KEK Internal　2006－1
June 2006
MD

PF におけるビームライン制御の標準化

Standardization of Beam Line Control Systems at the Photon Factory

放射光科学研究施設
ビームライン制御標準化 WG



High Energy Accelerator Research Organization

High Energy Accelerator Research Organization (KEK), 2006

KEK Reports are available from:

Science Information and Library Services Division
High Energy Accelerator Research Organization (KEK)
1-1 Oho, Tsukuba-shi
Ibaraki-ken, 305-0801
JAPAN

Phone: +81-29-864-5137
Fax: +81-29-864-4604
E-mail: irdpub@mail.kek.jp
Internet: http://www.kek.jp

目次

WG メンバー

伊藤健二　放射光第一系）
澤　博　放射光第二系）
小菅 隆　放射光第一系）
濁川和幸　放射光第一系）
森 丈晴　放射光第一系）
五十嵐教之　放射光第二系）
大田浩正　(三菱電機システムサービス 株))
渡邉一樹　(三菱電機システムサービス 株))
足立伸一　放射光第二系）

1.　はじめに

放射光研究施設（PF）では PF2.5GeV リングと PF-AR を含めておよそ 70 の実験ステーションがあり、放射光を利用した様々な分野の研究が行われている。これらの実験ステーションで測定を行うためにはその上流側に設置されている集光光学系、単色計、スリットなどの光学素子の駆動を制御するシステムが必要である。従来、これらの制御システム構築および運営は、ビームライン担当者の裁量に任されており、現存するシステムは非常にバライティーに富んでいる。これらの制御システムは、実験ステーションで行われる実験に合わせて構築されている点は評価されるが、１）ビームライン担当者が個々にシステムを構築することに費やすマンパワーの浪費、２）ビームライン担当者の交代において例えマニュアルが整備されていてもスムーズな引継ぎには問題が生じる可能性、３）システムの老朽化に伴う制御機器更新が必要となった時には、またビームライン毎にマンパワーの浪費、４）放射光を利用するユーザーの立場からみると、同じような実験を行う場合ですら、実験ステーションが異なると別の制御システムの操作方法を習得しなければならないこと、のような問題点がある。これらを解決する方法としては、PF における標準的な制御システムを構築しさらにその維持管理を専門的に行う BL 制御グループを PF 内に設けることが考えられる。

現在 PF で使用されている制御システムは多彩であるが、実際ビームラインでの制御対象要素は限られており、またそれらの駆動あるいはそれに用いられる機器についても共通性は高く、標準的な制御システムを構築することにそれほど大きな障壁はないように思われる。上記の BL 制御グループは、１）標準的な制御システムの開発、最新の制御技術のフォローアップ、２）ビームライン制御システムの新規構築あるいは更新時のビームライン担当者へアドバイス、３）ビームライン制御システムの構築作業、４）不具合が生じた時の早い対応、を行うことが考えられる。このことにより、ビームライン担当者はビームライン制御システムに関する仕事量の大部分が軽減されることが見込める。また、このように構築されるシステムはかなりレベルの高い up-date なものが期待される。これはかなり理想的なイメージであるが、ここまで行き着くまでには多くの問題点があるであろう。一つは、「標準的なビームライン制御システム」に対するコンセンサスの形成であろう。そこで、ビームライン制御システム標準化の作業グループを作り、PF で使用されているシステムの現状把握、共通化可能な部分および標準の設定について検討することにした。本冊子は、この WG での検討結果のまとめである。詳細は以下に譲るとして、本 WG では、「一般的な測定にはビームラインで用意した常設システムで対応し、高度な測定にはユーザー持込 PC を接続できる簡易アプリケーション間送受信システム（STARS）を基本とする」ことをベースに検討を進めた。

上述のように PF ではおよそ７０の実験ステーションが存在しており、これら全てを対象とした標準化を進めることは非常に難しい。しかし、幸いにも部分的にではあるが、ビームラインの再構築が進められており、ここでは本冊子で提案されている制御システムを採用していただけるであろう。また、ここで紹介している考え方は、必ずしも PF2.5GeV リングと PF-AR だけに留まることなく今後の開発作業も視野に入れ、将来計画として現在検討が始められた ERL でのビームライン制御に大いに役立つものにしていただきたい。

伊藤　健二

2. 標準化の指針

2.1 基本的な考え方

放射光実験を行う際には、ビームラインの上流から下流まで数多くのコンポーネントの状態を協調的に計測・制御しなければならない。制御対象となるコンポーネントの状態とは、アンジュレータやウィグラーのギャップ値、基幹チャンネルのシャッター・アブソーバ等のステータス、光学コンポーネントとして分光器・グレーティング・ミラー・光モニター・スリットなどの設定、実験ハッチ内の各種実験装置、ビームライン全体を司るインターロック制御システム、ビームラインの真空度や冷却水のステータス、蓄積リングの運転情報など、きわめて多岐にわたる。またこれらのコンポーネントは未来永劫同じ物が使用されるわけではなく、蓄積リングやビームラインのアップグレードに伴って絶えず更新されてゆく。このような「進化し続けるビームラインの計測・制御システム」を作り上げてゆくためには、はじめから全体の制御システムを１個のかたまりとして開発するのではなく、コンポーネントごとに分類して、個別システムを開発・統合する「分散化」の考え方が重要である。

上記のような視野に立って、PF では制御グループの小菅氏が中心となって、簡易アプリケーション間送受信システム STARS (Simple Transmission and Retrieval System)が開発された。STARS はネットワーク上でアプリケーション間の通信を司る「交通整理のためのクライアント・サーバープログラム」であり、以下のような特長を持つ。

1. アプリケーション間通信は、全て TCP/IP ソケットを利用したテキストベースのコマンドの送受により行われる。
2. 開発言語や OS の選択の幅が広い。
3. デバッグを行う場合にも TELNET などのツールを利用することが可能。
4. コアの部分のプログラムは Perl を使って開発されており、様々なプラットフォーム上で動作可能。

本ワーキンググループは、この STARS を制御システムの基盤として、放射光実験の制御システムを標準化するための仕様の作成を行った。

2.2 標準化されたシステムの構成

図１に STARS を使用したクライアント・サーバー制御システムの構成図を示す。STARS サーバーに接続されているクライアントのうち、ユーザー側のクライアントをユーザークライアント、装置側のクライアントを I/O クライアントと呼ぶ。個別コンポーネントの I/O クライアントとユーザークライアントを個別に開発し、それらを STARS サーバーを介して有機的に結合させることで、分散化した計測・制御システムの開発が可能となる。制御対象となる装置は、個々のビームラインによりその仕様がまちまちであるが、その標準化を行うためには、使用頻度の高い装置について、まず重点的にサポートを行うことが重要である。使用頻度を調査するために行ったアンケートの結果を参考資料（４．４）に示す。

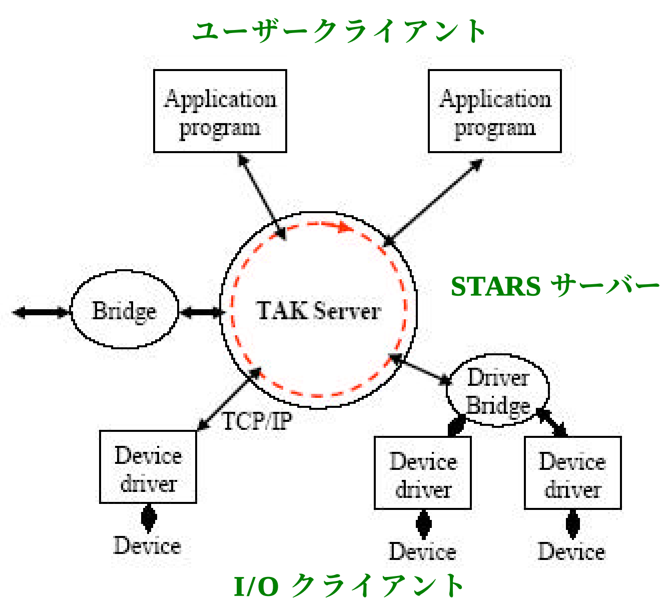


図 1: STARS の構成

アンケートの結果、第 1 段階として以下のデバイスおよび I/O クライアントを STARS 標準としてサポートすることとした。また他のデバイスについても、ビームラインでの使用頻度により、今後サポートされるデバイスが増加する予定である。

プログラム名	サポート対象			備考
	メーカー	型番	仕様	
WE7000	**YOKOGAWA**	**WE7562**	MCA モジュール	BL20
		WE7521	4ch タイミングカウンターモジュール	
		WE7262	32bit デジタル I/O モジュール	
pm16c04	**TSUJI DENSHI**	**PM16C-04S**	16ch4 台同時駆動パルスモータコントローラ	BL1A
		PM16C-04	PM16C-04S の Net 対応版	コラボ検証機
pm16c02		**PM16C-02Z**	16ch2 台同時駆動パルスモータコントローラ	BL1B
npm2c01		**NPM2C-01**	2ch パルスモータコントローラNet 対応廉価版	近日リリース
sc400	**KOHZU**	**SC-200**	2ch パルスモータコントローラ	構造生物 G
		SC-400	4ch パルスモータコントローラ	実機未検証
		SC-800	8ch パルスモータコントローラ	実機未検証
m2701drv	**KEITHLEY**	**Model 2000-SCAN**	デジタル マルチメータ (6.5 桁) 10 チャンネルスキャナカード	BL1B
		Model 2701+7700	デジタルマルチメータ (6.5 桁、22 ビット積分型 AD、Net 対応) +7700 (プラグインスイッチ / 測定モジュー ル、20ch、差動入力)	BL1A

・近日中にサポート予定のデバイス（および I/O クライアント）

TWM5501（パルスコントローラモジュール）　太陽計測

974、994、995（カウンタ・タイマ）　Ortec

3. 制御コマンドの仕様

3. 1 STARS で制御をサポートするビームラインコンポーネント

2 章で掲げた STARS 標準デバイスを採用しているビームラインコンポーネントについて、STARS サーバー経由で制御するための共通 I/O クライアント（および GUI)が、制御グループによってすでに作成されています。

・標準ビームラインコンポーネントリスト

- ・分光器： energy (eV), wavelength (angstrom), angle (degree)
- ・スリット： width (mm), height (mm), horizontal position (mm), vertical position (mm)
- ・ミラー： (1 枚物) x, y, z (mm), thetax, thetay, thetaz (mrad) （実軸または仮想軸）
  - (2 枚物) 面間隔 d (mm)
- ・ピエゾ： dtheta1 (arcsec)
- ・パルスモーター： 各軸 (pulse)
- ・アンジュレータ： gap (mm), taper (mm)
- ・蓄積電流値： mA (データ取得のみ)
- ・MBS： ステータス（データ取得のみ)
- ・BBS： ステータス（データ取得のみ)
- ・入射予告： 次回入射までの時間（データ取得のみ)

3. 2 基本的な仕様

※ 実際にユーザーが接続することを意識してSTARS のI/O クライアントと言わず、ビームラインコンポーネントと表現しました。

コマンド

基本的なコマンドは値の取得(GetValue)､値のセット(SetValue)､駆動中であるかの問い合わせ(IsBusy)､駆動の停止(Stop)からなります。コマンドはビームラインコンポーネント(実際にはSTARSのClient)の名前を指定して“ コンポーネント名 コマンド ”のように送出する事となります。ただし、それぞれのビームラインコンポーネント名はピリオドを使用することによって階層化されたイメージで表現されるので、たとえば分光器のエネルギーの場合であれば“ Mono.Energy ”のように表現されます。

例： Mono.Energy GetValue (分光器のエネルギーを取得)
また、コマンドを送出すると必ず“ @ ”で始まるリプライメッセージが
“ コンポーネント＞接続時に指定したターミナル名? @コマンド? 結果 ”
のように送られてきます。以下の例はターミナル名を“ user1 ”として接続した場合の上の例に対する回答です。
例: Mono.Energy>user1 @GetValue xxxx (モノクロのエネルギーがxxxx eV)

イベント

あらかじめ、“ System flgon コンポーネント名 ”のようなコマンドを送出すると指定したコンポーネントの状態が変化したときに“ イベントメッセージ ”を受け取る事が可能です。一度System flgon コ

マンドを送出すると、そのコンポーネントの状態が変化するたびに“コンポーネント名>接続時に指定したターミナル名”に加えて“__”（アンダースコア）から始まるイベントメッセージが送られてきます。基本的なイベントとしては値が変化した時のイベント(_ChangedeValue)及び駆動が開始され等でStop コマンド以外を受け取らない事を示すためのイベント(_ChangedIsBusy)があります。
なお、イベントの受け取りをとりやめたい場合には“System flgoff コンポーネント名”のようにフラグオフコマンドを送信します。以下は分光器の角度に対してflgon したときの例です。
例：分光器の角度の状態が変わったらイベントを受け取る(user1 のターミナル名で接続した場合)
System flgon Mono.angle (分光器の角度の状態が変わったらイベントを受け取る)
System>user1 @flgon Node Mono.angle has been registered. (リプライメッセージ)
：
Mono.angle _ChangedIsBusy 1 (分光器が駆動された)

利用例

以下に分光器のエネルギーについてのコマンド及びイベントを例に示します。ここでEnergy は分光器に属するものとしているのでコンポーネント名は“Mono.Energy”のように表現されます。
Mono.Energy GetValue（分光器エネルギーの取得）
分光器のエネルギーを取得します。単位は“eV”です。
例（ターミナル名user1 として接続した場合）：
Mono.Energy GetValue
Mono.Energy>user1 @GetValue xxxx (分光器のエネルギーはxxxx eV)
Mono.Energy IsBusy（分光器が駆動中であるかの問い合わせ）
分光器が駆動中であるかの問い合わせを行います。駆動中であれば1 が、停止していて駆動が可能であれば0 を返します。
例（ターミナル名user1 として接続した場合）：
Mono.Energy IsBusy (分光器が駆動中であるか?)
Mono.Energy>user1 @IsBusy 1 (分光器が駆動中である)
Mono.Energy SetValue value（分光器エネルギーをセット）
分光器のエネルギーを“eV”を単位としてセットします。
例（ターミナル名user1 として接続した場合）：
Mono.Energy SetValue xxxx (分光器のエネルギーをxxxx eV にセット)
Mono.Energy>user1 @SetValue xxxx Ok: (コマンドを受付け、駆動を開始)
Mono.Energy Stop（分光器駆動の停止）
駆動中である分光器を停止させます。
例（ターミナル名user1 として接続した場合）：
Mono.Energy Stop
Mono.Energy>user1 @Stop Ok: (停止コマンドを受け取った)
イベント
イベントとしては“_ChangedIsBusy”（分光器が駆動あるいは停止した場合）、“_ChangedValue”（エネルギーの値が変わった場合）があり、あらかじめ“System flgon Mono.Energy”コマンドを送ると送信されてきます。

例（ターミナル名user1 **として接続した場合）：**

System flgon Mono.Energy (**分光器のエネルギーの状態が変わったらイベントを受け取る**)

System>user1 @flgon Node Mono.angle has been registered. (**リプライメッセージ**)

Mono.Energy SetValue xxxx (**分光器のエネルギーを**xxxx eV **にセット**)

Mono.Energy>user1 @SetValue xxxx Ok: (**コマンドを受付け、駆動を開始**)

Mono.Energy>user1 _ChangedIsBusy 1 (**分光器が駆動を開始した**)

Mono.Energy>user1 _ChangedValue yyyy (**分光器が駆動中で経過として**yyyy **に変化**)

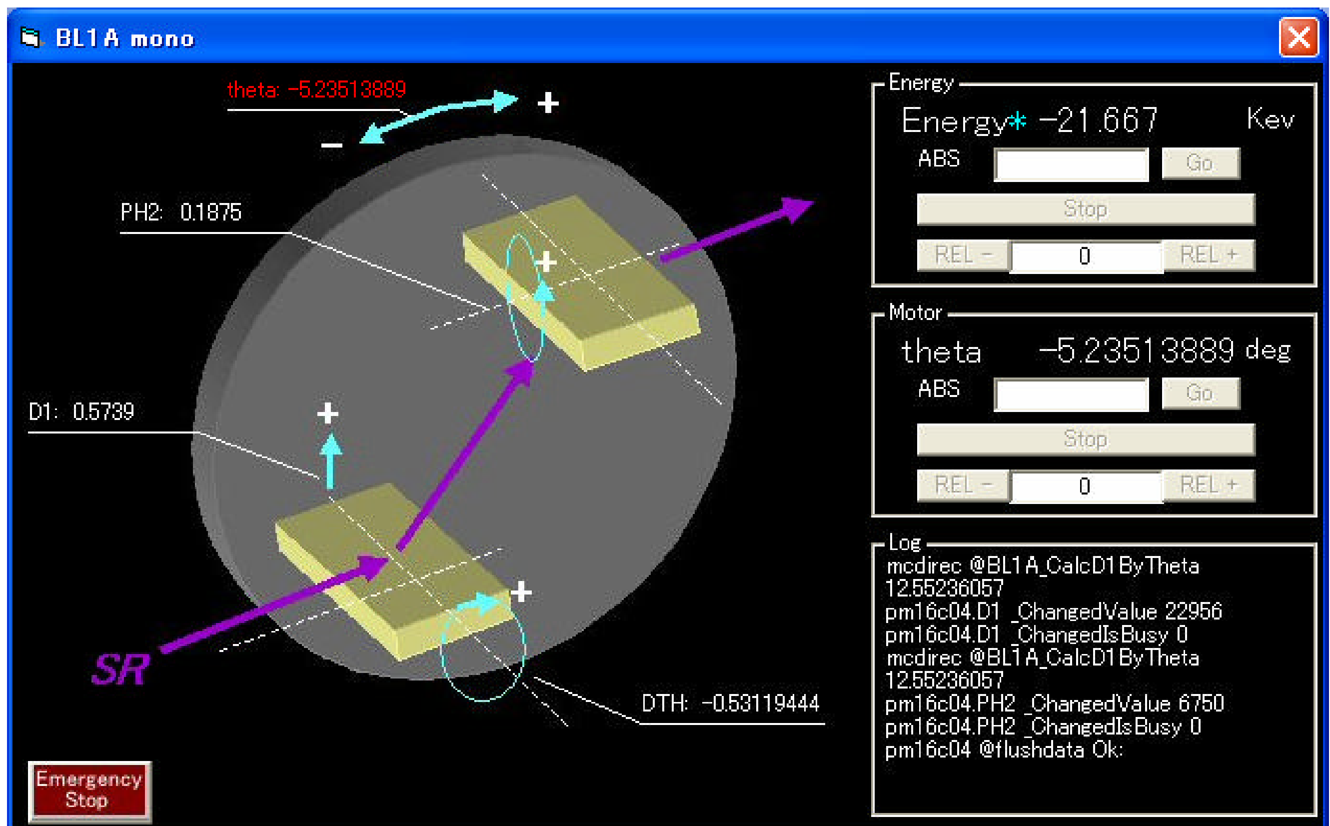
：目的値に達するまで同様のイベントが数秒程度の間隔で送られてくる。

Mono.Energy>user1 _ChangedValue xxxx (**分光器が目的地まで移動**)

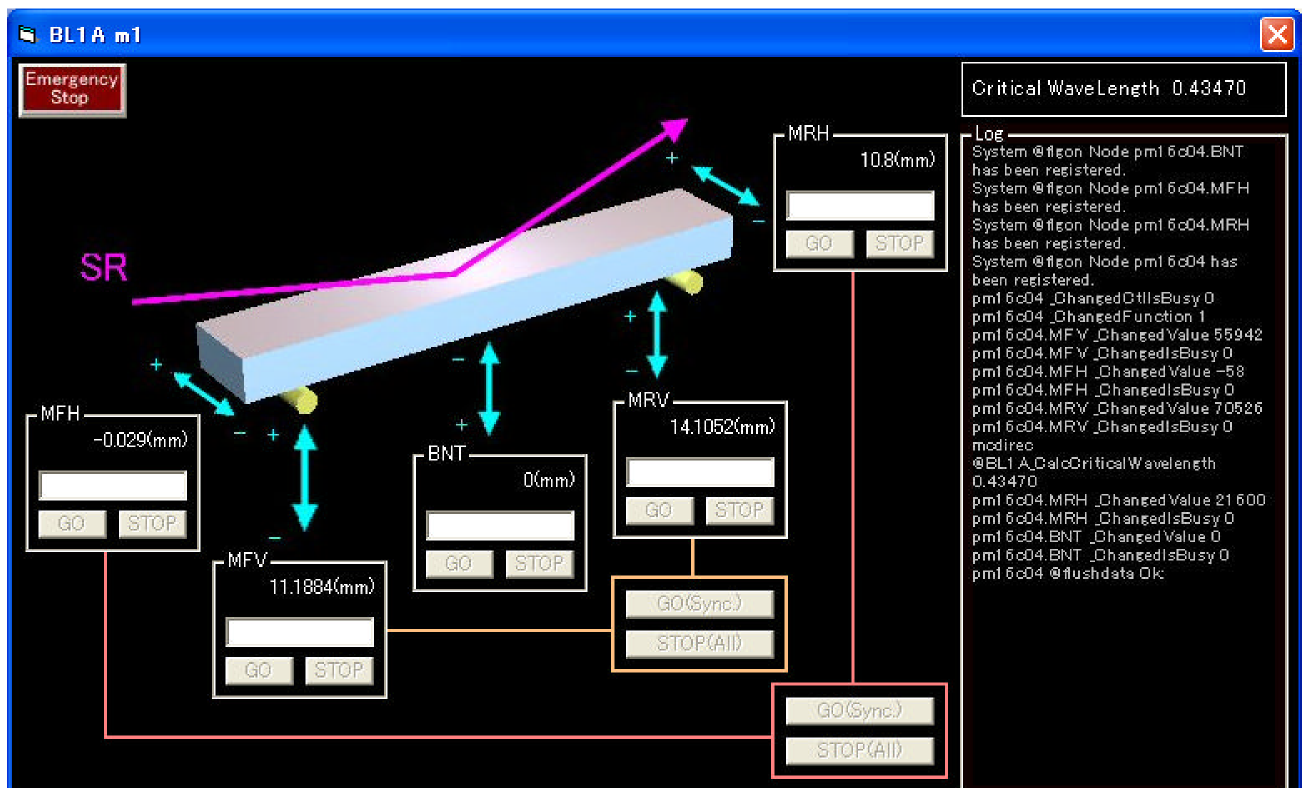
Mono.Energy>user1 _ChangedIsbusy (**分光器が停止**)

3.3 ビームラインコンポーネント制御用 GUI の例

3.3.1 モノクロメータ制御用 GUI



3.3.2 集光ミラー制御用 GUI



3. 3. 3.　スキャン用GUI

３．３．４　デスクトップ例

3. 4 各コンポーネントのコマンド仕様

BLコンポーネント	プロパティー	コマンド/ イベント例 (User Client 名が"user"の場合	意味
分光器(Mono)	Energy	Mono.Energy GetValue	分光器のエネルギーの値をeV で取得
		Mono.Energy IsBusy	分光器のエネルギーの状態を 1(駆動中)または 0(停止中)で取得する
		Mono.Energy SetValue xxxx	分光器のエネルギーを xxxx eV にセットする
		Mono.Energy Stop	分光器のエネルギーの駆動を中止(停止)する
		Mono.Energy>user1 _ChangedValue xxxx	分光器のエネルギーの値が xxxx eV に変化した事が通知される
		Mono.Energy>user1 _ChangedIsBusy (1 or 0)	分光器のエネルギーが駆動開始(1)または停止(0)した事が通知される
	Wavelength	Mono.Wavelength GetValue	分光器の波長の値をangstrom で取得
		Mono.Wavelength IsBusy	分光器の波長の状態を 1(駆動中)または 0(停止中)で取得する
		Mono.Wavelength SetValue xxxx	分光器の波長を xxxx angstrom にセットする
		Mono.Wavelength Stop	分光器の波長の駆動を中止(停止)する
		Mono.Wavelength>user1 _ChangedValue xxxx	分光器の波長の値が xxxx angstrom に変化した事が通知される
		Mono.Wavelength>user1 _ChangedIsBusy (1 or 0)	分光器の波長が駆動開始(1)または停止(0)した事が通知される
	Angle	Mono.Angle GetValue	分光器の角度の値をdegree で取得
		Mono.Angle IsBusy	分光器の角度の状態を 1(駆動中)または 0(停止中)で取得する
		Mono.Angle SetValue xxxx	分光器の角度を xxxx degree にセットする
		Mono.Angle Stop	分光器の角度の駆動を中止(停止)する

		Mono.Angle>user1 _ChangedValue xxxx	分光器の角度の値が xxxx degree に変化した事が通知される
		Mono.Angle>user1 _ChangedIsBusy (1 or 0)	分光器の角度が駆動開始(1)または停止(0)した事が通知される
スリット1(Slit1)	Width	Slit1.Width GetValue	スリット1 の幅の値をmm で取得
		Slit1.Width IsBusy	スリット1 の幅の状態を 1(駆動中)または 0(停止中)で取得する
		Slit1.Width SetValue xxxx	スリット1 の幅を xxxx mm にセットする
		Slit1.Width Stop	スリット1 の幅の駆動を中止(停止)する
		Slit1.Width>user1 _ChangedValue xxxx	スリット1 の幅の値が xxxx mm に変化した事が通知される
		Slit1.Width>user1 _ChangedIsBusy (1 or 0)	スリット1 の幅が駆動開始(1)または停止(0)した事が通知される
	Height	Slit1.Height GetValue	スリット1 の高さの値をmm で取得
		Slit1.Height IsBusy	スリット1 の高さの状態を 1(駆動中)または 0(停止中)で取得する
		Slit1.Height SetValue xxxx	スリット1 の高さを xxxx mm にセットする
		Slit1.Height Stop	スリット1 の高さの駆動を中止(停止)する
		Slit1.Height>user1 _ChangedValue xxxx	スリット1 の高さの値が xxxx mm に変化した事が通知される
		Slit1.Height>user1 _ChangedIsBusy (1 or 0)	スリット1 の高さが駆動開始(1)または停止(0)した事が通知される
	HPosition	Slit1.HPosition GetValue	スリット1 の水平位置の値をmm で取得
		Slit1.HPosition IsBusy	スリット1 の水平位置の状態を 1(駆動中)または 0(停止中)で取得する
		Slit1.HPosition SetValue xxxx	スリット1 の水平位置を xxxx mm にセットする
		Slit1.HPosition Stop	スリット1 の水平位置の駆動を中止(停止)する
		Slit1.HPosition>user1 _ChangedValue xxxx	スリット1 の水平位置の値が xxxx mm に変化した事が通知される
		Slit1.HPosition>user1 _ChangedIsBusy (1 or	スリット1 の水平位置が駆動開始

		0)	(1)または停止(0)した事が通知される
	VPosition	Slit1.VPosition GetValue	スリット1 の垂直位置の値をmm で取得
		Slit1.VPosition IsBusy	スリット1 の垂直位置の状態を 1(駆動中)または 0(停止中)で取得する
		Slit1.VPosition SetValue xxxx	スリット1 の垂直位置を xxxx mm にセットする
		Slit1.VPosition Stop	スリット1 の垂直位置の駆動を中止(停止)する
		Slit1.VPosition>user1 _ChangedValue xxxx	スリット1 の垂直位置の値が xxxx mm に変化した事が通知される
		Slit1.VPosition>user1 _ChangedIsBusy (1 or 0)	スリット1 の垂直位置が駆動開始(1)または停止(0)した事が通知される
ミラー1(Mirr1)	X	Mirr1.X GetValue	ミラー1 の X 位置の値をmm で取得
		Mirr1.X IsBusy	ミラー1 の X 位置の状態を 1(駆動中)または 0(停止中)で取得する
		Mirr1.X SetValue xxxx	ミラー1 の X 位置を xxxx mm にセットする
		Mirr1.X Stop	ミラー1 の X 位置の駆動を中止(停止)する
		Mirr1.X>user1 _ChangedValue xxxx	ミラー1 の X 位置の値が xxxx mm に変化した事が通知される
		Mirr1.X>user1 _ChangedIsBusy (1 or 0)	ミラー1 の X 位置が駆動開始(1)または停止(0)した事が通知される
	Y	Mirr1.Y GetValue	ミラー1 の Y 位置の値をmm で取得
		Mirr1.Y IsBusy	ミラー1 の Y 位置の状態を 1(駆動中)または 0(停止中)で取得する
		Mirr1.Y SetValue xxxx	ミラー1 の Y 位置を xxxx mm にセットする
		Mirr1.Y Stop	ミラー1 の Y 位置の駆動を中止(停止)する
		Mirr1.Y>user1 _ChangedValue xxxx	ミラー1 の Y 位置の値が xxxx mm に変化した事が通知される
		Mirr1.Y>user1 _ChangedIsBusy (1 or 0)	ミラー1 の Y 位置が駆動開始(1)または停止(0)した事が通知される
	Z	Mirr1.Z GetValue	ミラー1 の Z 位置の値をmm で取得

		Mirr1.Z IsBusy	ミラー1 の Z 位置の状態を 1(駆動中)または 0(停止中)で取得する
		Mirr1.Z SetValue xxxx	ミラー1 の Z 位置を xxxx mm　にセットする
		Mirr1.Z Stop	ミラー1 の Z 位置の駆動を中止(停止)する
		Mirr1.Z>user1 _ChangedValue xxxx	ミラー1 の Z 位置の値が xxxx mm に変化した事が通知される
		Mirr1.Z>user1 _ChangedIsBusy (1 or 0)	ミラー1 の Z 位置が駆動開始(1)または停止(0)した事が通知される
	ThetaX	Mirr1.ThetaX GetValue	ミラー1 の θX の値をmrad で取得
		Mirr1.ThetaX IsBusy	ミラー1 の θX の状態を 1(駆動中)または 0(停止中)で取得する
		Mirr1.ThetaX SetValue xxxx	ミラー1 の θX を xxxx mrad　にセットする
		Mirr1.ThetaX Stop	ミラー1 の θX の駆動を中止(停止)する
		Mirr1.ThetaX>user1 _ChangedValue xxxx	ミラー1 の θX の値が xxxx mrad に変化した事が通知される
		Mirr1.ThetaX>user1 _ChangedIsBusy (1 or 0)	ミラー1 の θX が駆動開始(1)または停止(0)した事が通知される
	ThetaY	Mirr1.ThetaY GetValue	ミラー1 の θy の値をmrad で取得
		Mirr1.ThetaY IsBusy	ミラー1 の θy の状態を 1(駆動中)または 0(停止中)で取得する
		Mirr1.ThetaY SetValue xxxx	ミラー1 の θy を xxxx mrad　にセットする
		Mirr1.ThetaY Stop	ミラー1 の θy の駆動を中止(停止)する
		Mirr1.ThetaY>user1 _ChangedValue xxxx	ミラー1 の θy の値が xxxx mrad に変化した事が通知される
		Mirr1.ThetaY>user1 _ChangedIsBusy (1 or 0)	ミラー1 の θ yが駆動開始(1)または停止(0)した事が通知される
	ThetaZ	Mirr1.ThetaZ GetValue	ミラー1 の θz の値をmrad で取得
		Mirr1.ThetaZ IsBusy	ミラー1 の θz の状態を 1(駆動中)または 0(停止中)で取得する
		Mirr1.ThetaZ SetValue xxxx	ミラー1 の θz を xxxx mrad　にセットする
		Mirr1.ThetaZ Stop	ミラー1 の θz の駆動を中止(停止)する

		Mirr1.ThetaZ>user1 _ChangedValue xxxx	ミラー1 のθz の値が xxxx mrad に変化した事が通知される
		Mirr1.ThetaZ>user1 _ChangedIsBusy (1 or 0)	ミラー1 のθzが駆動開始(1)または停止(0)した事が通知される
Piezo 直接の場合	Piezo1	Piezo1 GetValue	ピエゾ1 の値をarcsec で取得
		Piezo1 IsBusy	ピエゾ1 の状態を 1(駆動中)または 0(停止中)で取得する
		Piezo1 SetValue xxxx	ピエゾ1 を xxxx arcsec にセットする
		Piezo1 Stop	ピエゾ1 の駆動を中止(停止)する
		Piezo1>user1 _ChangedValue xxxx	ピエゾ1 の値が xxxx arcsec に変化した事が通知される
		Piezo1>user1 _ChangedIsBusy (1 or 0)	ピエゾ1 が駆動開始(1)または停止(0)した事が通知される
Pulse Motor 直接の場合	PMotor1	PMotor1 GetValue	パルスモータ1 の値をpulse で取得
		PMotor1 IsBusy	パルスモータ1 の状態を 1(駆動中)または 0(停止中)で取得する
		PMotor1 SetValue xxxx	パルスモータ1 を xxxx pulse にセットする
		PMotor1 Stop	パルスモータ1 の駆動を中止(停止)する
		PMotor1>user1 _ChangedValue xxxx	パルスモータ1 の値が xxxx pulse に変化した事が通知される
		PMotor1>user1 _ChangedIsBusy (1 or 0)	パルスモータ1 が 1(駆動中)または 0(停止中)した事が通知される
挿入光源(ID)	Gap	ID.Gap GetValue	挿入光源のギャップの値をmm で取得
		ID.Gap IsBusy	挿入光源のギャップの状態を mm で取得する
		ID.Gap SetValue xxxx	挿入光源のギャップを xxxx mm にセットする
		ID.Gap Stop	挿入光源のギャップの駆動を中止(停止)する
		ID.Gap>user1 _ChangedValue xxxx	挿入光源のギャップの値が xxxx mm に変化した事が通知される
		ID.Gap>user1 _ChangedIsBusy (1 or 0)	挿入光源のギャップが駆動開始(1)または停止(0)した事が通知される

	Taper	ID.Taper GetValue	挿入光源のテーパーの値をmm で取得
		ID.Taper IsBusy	挿入光源のテーパーの状態を1(駆動中)または 0(停止中)で取得する
		ID.Taper SetValue xxxx	挿入光源のテーパーを xxxx mm にセットする
		ID.Taper Stop	挿入光源のテーパーの駆動を中止(停止)する
		ID.Taper>user1 _ChangedValue xxxx	挿入光源のテーパーの値が xxxx mm　に変化した事が通知される
		ID.Taper>user1 _ChangedIsBusy (1 or 0)	挿入光源のテーパーが駆動開始(1)または停止(0)した事が通知される
リング情報(Ring)	DCCT	Ring.DCCT GetValue	リング情報の蓄積電流の値をmAで取得
	Lifetime	Ring.Lifetime GetValue	リング情報のライフタイムの状態をminutes で取得する
	toNextInjection	Ring.toNextInjection GetValue	リング情報の次回入射までの時間を xxxx seconds　にセットする
インターロック情報(BLIS)	MBS	BLIS.MBS GetValue	インターロック情報の MBS の駆動を中止(停止)する
	BBS	BLIS.BBS GetValue	インターロック情報の BBS の値が xxxx Open/Closed/Neutral/Error に変化した事が通知される
	DSS	BLIS.DSS GetValue	インターロック情報の DSS が Open/Closed/Neutral/Error した事が通知される

４．参考資料

４．１　PM16C04 STARS I/O Client コマンド概要

2005-12-13

はじめに

ツジ電子製PM16C04は16台のパルスモータを同時に4台まで駆動可能なパルスモータコントローラですが、「チャンネルにモータを選んで駆動する」といった動作が必要となるために若干扱いが面倒です。このSTARSクライアント(I/O Client)を使うとチャンネルの選択はpm16c04 Client側で自動で行ってくれるので、ユーザは「どのモータを選んで」などという事を考える必要がなくなります。

PM16C04 STARS I/O Client(以下、単にPM16C04とします)の作りはコントローラの先にモータが接続されているとような形をイメージしていているので、各モータへの命令は、送り先を"PM16C4.dth1"のように階層化したような形式にします。

```
  -------      (コントローラ)   (モータ名)
 |STARS  |------PM16C04---------th
 | Server|                 |
  -------                  +----dth1
                           |
                           +----all
                           :
                           : 略
```

例： term1というUser ClientからPM16C04に接続されたdth1というモータのポジションを取得
[term1から]
PM16C04.dth1 GetValue

[PM16C04からの返事]
PM16C04.dth1>term1 @GetValue 12345

また、コントローラであるPM16C04自身もコマンドを持っていますのでそれにアクセスする場合には

[term1から PM16Cの状態(Remote or Local)の取得]
PM16C04 GetFunction

[PM16C04からの返事]
PM16C04>term1 @GetFunction 1

のように送受信します。

以下、ここではこの PM16C04 のコマンドの概要について説明します。

主要コマンドおよびイベント

モータへのアクセスは、"はじめに"で述べたとおり"PM16C04.dth1"のように行うわけですが、実は PM16C04 自体のコマンドの中にもモータ関係のコマンドがあり、

PM16C04 GetValue 15 (モータ番号 15 のポジションを取得)

とすることも可能です。しかし、"主要コマンド"ではこれらのコマンドについては触れず、モータを動かす上で最低限必要と思われるものについて説明します。なお、Example:はいずれも"term1"と言う名前を持った User Client からのコマンド送信を例としています。

GetValue

Usage: GetValue
Synopsis: モータの現在値を取得。
Reply: @GetValue *Value*
Error: @GetValue Er: *Reason*

Example:
PM16C04.dth1 GetValue (term1 から送信)
PM16C04.dth1>term1 @GetValue 12345 (受信)

PM16C04.dth1 GetValue
PM16C04.dth1>term1 @GetValue Er: Timeout (コントローラの電源が OFF だった等でタイムアウトした)

SetValue

Usage: SetValue *Value*
Synopsis: モータを Value の値まで動かします。

Reply:　　@SetValue *Value* Ok:
Error:　　@SetValue Er: *Reason*
Example:
PM16C04.dth1 SetValue 12345 (term1から送信)
PM16C04.dth1>term1 @SetValue 12345 Ok: (受信)

PM16C04.dth1 SetValue 8388608
PM16C04.dth1>term1 @SetValue 8388608 Er: Data out of range. (範囲外のためエラー)

SetValueを実行すると対象となるモータが動作中、あるいはコントローラがBusy状態である、コントローラが"Local"であるような場合以外は例で示すように"Ok:"をすぐに返します。モータが止まったことを検知するためには"IsBusy"を使って問い合わせるか、あらかじめ

System flgon PM16C04.dth1 (term1からSystemにイベント配信要求)
System>term1 @flgon Node PM16C04.dth1 has been registered. (STARSサーバーからの回答)

のようにして、モータの停止後

PM16C04.dth1>term1 _ChangedIsBusy 0 (dth1が停止したというイベントを受信)

のような、イベントを受け取るようにします。

Preset

Usage:　　Preset *Value*
Synopsis: モータの現在値をプリセットします。
Reply:　　@Preset *Value* Ok:
Error:　　@Preset Er: *Reason*
Example:
PM16C04.dth1 Preset 12345 (term1から送信)
PM16C04.dth1>term1 @Preset 12345 Ok: (受信)

PM16C04.dth1 Preset 8388608
PM16C04.dth1>term1 @Preset 8388608 Er: Data out of range. (範囲外のためエラー)

IsBusy

Usage:　IsBusy

Synopsis: モータが駆動可能であるかの問い合わせ。1=Busy、0=駆動可能

Reply:　@IsBusy *Value*

Example:

PM16C04.dth1 IsBusy（term1から送信）

PM16C04.dth1>term1 @IsBusy 1（受信：動作中である）

Stop

Usage:　Stop

Synopsis: モータを台形駆動の設定に沿って減速停止させる。

Reply:　@Stop Ok:

Example:

PM16C04.dth1 Stop（term1から送信）

PM16C04.dth1>term1 @Stop Ok:（受信）

StopEmergency

Usage:　StopEmergency

Synopsis: モータを台形駆動の設定に関係なく瞬時に停止させる。

Reply:　@StopEmergency Ok:

Example:

PM16C04.dth1 StopEmergency（term1から送信）

PM16C04.dth1>term1 @StopEmergency Ok:（受信）

Standby(コントローラのコマンド)

Usage:　Standby

Synopsis: 同時駆動のために全チャンネルをStandby状態にする。

Reply:　@Standby Ok:

Error:　@Standby Er: *Reason*

Example:

PM16C04 Standby (term1 から送信)
PM16C04>term1 @Standby Ok: (受信)
PM16C04.m1u SetValue 12345 (m1u を 12345 へ、まだ駆動は開始しない)
PM16C04.m1u>term1 @SetValue 12345 Ok: (受信)
PM16C04.m1d SetValue 12345 (m1d を 12345 へ、まだ駆動は開始しない)
PM16C04.m1d>term1 @SetValue 12345 Ok: (受信)
PM16C04 SyncRun (駆動開始))
PM16C04>term1 @SyncRun Ok: (受信)

SyncRun(コントローラのコマンド)

Usage:　　SyncRun
Synopsis: 同時駆動の開始。
Reply:　　@SyncRun Ok:
Error:　　@SyncRun Er: *Reason*
Example:　Standby の項を参照

_ChangedIsBusy イベント

Event:　　_ChangedIsBusy *Value*
Synopsis: PM16C04 配下のモータが駆動を開始したときや停止したときには STARS サーバー(System)に
その旨を示すイベントが送信されます。
"System flgon PM16C04.dth1"のようにあらかじめ設定することで User Client で
受信が可能となります。モータが動き始めた場合には Value が 1、停止した場合には Value が 0
として送られてきます。
Example:
System flgon PM16C04.dth1 (term1 から flgon)
System>term1 @flgon Node pm16c04.th has been registered. (受信)
PM16C04.dth1 SetValue 1000 (term1 から dth1 に SetValue コマンド)
PM16C04.dth1>term1 _ChangedIsBusy 1 (dth1 が駆動開始したというイベントを受信)
PM16C04.dth1>term1 @SetValue 1000 Ok:
PM16C04.dth1>term1 _ChangedValue 351 (Value が変化した事を示すイベントを受信)
PM16C04.dth1>term1 _ChangedValue 899
PM16C04.dth1>term1 _ChangedValue 999
PM16C04.dth1>term1 _ChangedValue 1000
PM16C04.dth1>term1 _ChangedIsBusy 0 (dth1 が停止した事を示すイベントを受信)

_ChangedValue イベント

Event:　　_ChangedValue *Value*
Synopsis: PM16C04 配下のモータの Value が変化すると STARS サーバーに現在の Value データが
　　　　　イベントととして定期的に送られます。
　　　　　"System flgon PM16C04.dth1"のようにあらかじめ設定することで User Client で
　　　　　受信が可能となります。
Example:　_ChangedIsBusy イベントの項を参照

_ChangedCtlIsBusy イベント(コントローラ)

Event:　　_ChangedCtlIsBusy *Value*
Synopsis: PM16C04 配下のモータのモータが同時に 4 台以上動きはじめこれ以上他のモータを動かせなく
　　　　　なった場合(Busy)、あるいは 4 つのうちのモータのどれかが停止しし、駆動命令受付が可能と
　　　　　なった場合(Free)、コントローラからイベントが STARS サーバーに送信されます。
　　　　　"System flgon PM16C04.dth1"のようにあらかじめ設定することで User Client で
　　　　　受信が可能となります。Busy になった場合には Value が 1、Free の場合には Value が 0
　　　　　として送られてきます。
Example:
System flgon PM16C04 (term1 から flgon)
System>term1 @flgon Node PM16C04 has been registered. (System から受信)
PM16C04>term1 _ChangedCtlIsBusy 1 (同時にモータが 4 台駆動して Busy となった)
PM16C04>term1 _ChangedCtlIsBusy 0 (コントローラが Free となった。)

_ChangedFunction イベント(コントローラ)

Event:　　_ChangedFunction *Value*
Synopsis: PM16C04 の FUNCTION が REM(Remote)あるいは LOC(Local)に変化すると STARS サーバーにイベントが
　　　　　送出されます。
　　　　　"System flgon PM16C04"のようにあらかじめ設定することで User Client で
　　　　　受信が可能となります。Remote になった場合には Value が 1、Local の場合には Value が 0
　　　　　として送られてきます。
Example:

System ***flgon*** *PM16C04* (term1 から flgon)

*System>**term1** @flgon Node PM16C04 has been **registered**.* (System から受信)

*PM16C04>**term1** _ChangedFunction 0* (Local になった旨を示すイベントを受信)

*PM16C04>**term1** _ChangedFunction 1* (Remote になった旨を示すイベントを受信)

コマンド一覧

以下に PM16C04 STARS I/O Client の全コマンド及びイベントを示します。

#STARS Commands

Usage: help [Commmand]

Target: Controller, Motor

List commands or show usage (with "command")

Usage: hello

Target: Controller, Motor

The client returns "@hello nice to meet you."

Usage: flushdata

Target: Controller

Get all status of PM16C-04 and sends event messages to "System".

Usage: flushdatatome

Target: Controller

Get all status of PM16C-04 and sends event messages to me.

#Status read

Usage: GetAccRate MotorNumber

Target: Controller

Get acceleration rate of "MotorNumber" (0 to 16).

Usage: GetAccRate

Target: Motor

Get acceleration rate.

Usage: GetCancelBacklash MotorNumber

Target: Controller

Get cancel backlash of "MotorNumber" (0 to 16).

Usage: GetCancelBacklash

Target: Motor
    Get cancel backlash.

Usage: GetDigitalCcwLs MotorNumber
Target: Controller
    Get CCW software limit switch (DIGITAL LS) of "MotorNumber" (0 to 16).

Usage: GetDigitalCcwLs
Target: Motor
    Get CCW software limit switch (DIGITAL LS).

Usage: GetDigitalCwLs MotorName|MotorNumber
Target: Controller
    Get CW software limit switch (DIGITAL LS) of "MotorNumber" (0 to 16).

Usage: GetDigitalCwLs
Target: Motor
    Get CW software limit switch (DIGITAL LS).

Usage: GetFunction
Target: Controller
    Get function "1=Remote/0=Local".

Usage: GetFunctionStatus
Target: Controller
    Get limit switch and "remote/local" status on channel A and B.
    bit 0: C POS CW LS, 1: C POS CCW LS, 2: C POS Z. LS,
        4: D POS CW LS, 5: D POS CCW LS, 6: D POS Z. LS,
        8: A POS CW LS, 9: A POS CCW LS, A: A POS Z. LS, B: STATUS CPU/MANU
        C: B POS CW LS, D: B POS CCW LS, E: B POS Z. LS

Usage: GetHighSpeed MotorNumber
Target: Controller
    Get high speed value of "MotorNumber" (0 to 16).

Usage: GetHighSpeed
Target: Motor
    Get high speed value.

Usage: GetJogPulse MotorNumber

Target: Controller
    Get jog pulse value of "MotorNumber" (0 to 16).

Usage: GetJogPulse
Target: Motor
    Get jog pulse value.

Usage: GetLimits MotorNumber
Target: Controller
    Get limit switch value of "MotorNumber" (0 to 16) in register.
    bit 0: CW LS A/B, 1: CCW LS A/B, 2: Z LS A/B, 3: CW LS ENABLE
        4: CCW LS ENABLE, 5: DIGITAL LS ENABLE, 6: HOLD, 7: MOTOR OFF

Usage: GetLimits
Target: Motor
    Get limit switch value in register.
    bit 0: CW LS A/B, 1: CCW LS A/B, 2: Z LS A/B, 3: CW LS ENABLE
        4: CCW LS ENABLE, 5: DIGITAL LS ENABLE, 6: HOLD, 7: MOTOR OFF

Usage: GetLowSpeed MotorNumber
Target: Controller
    Get low speed value of "MotorNumber" (0 to 16).

Usage: GetLowSpeed
Target: Motor
    Get low speed value of "MotorName" or "MotorNumber" (0 to 16).

Usage: GetMiddleSpeed MotorNumber
Target: Controller
    Get middle speed value of "MotorNumber" (0 to 16).

Usage: GetMiddleSpeed
Target: Motor
    Get middle speed value.

Usage: GetValue Channel MotorNumber
Target: Controller
    Get postion data of "Channle" (A, B, C, D) or "MotorNumber" (0 to 16).

Usage: GetValue

Target: Motor
    Get postion data.

Usage: GetStatus Channel
Target: Controller
    Get status register value of "Channel" (A, B, C, D).
    bit 0: BUSY, 1: DRIVE, 2: not used, 3: not used
        4: COMERR, 5: LDEND, 6: SSEND, 7: ESEND

Usage: IsBusy Channel|MotorNumber
Target: Controller
    Check, is motor busy?

Usage: IsBusy
Target: Motor
    Check, is motor busy?

Usage: GetCtlIsBusy
Target: Controller
        Check, is controller busy?

Usage: GetSelected Channel|MotorNumber
Target: Controller
    Get selected channel (A, B, C, D or N = not selected) with
    "MotorName" or "MotorNumber" (0 to 16) or get selected motor number with
    "Channel" (A, B, C, D).

Usage: GetSelected
Target: Motor
    Get selected channel (A, B, C, D or N = not selected).

#PM16C Commands
Usage: GetRomVersion
Target: Controller
    Get firmware virsion of PM16C-04.

Usage: Standby
Target: Controller
    Standby motor(s). The "Standby" command is used for starting 2 motors at
    the same time with "SyncRun" command.

Usage: SyncRun
Target: Controller
    Start motor(s). The "SyncRun" command is used for starting 2 motors at
    the same time with "Standby" command.

Usage: Remote
Target: Controller
    Set function to "Remote". (Same as "SetFunction 1")

Usage: Local
Target: Controller
    Set function to "Local". (Same as "SetFunction 0")

Usage: SpeedLow
Target: Controller
    Set speed to "Low".

Usage: SpeedMiddle
Target: Controller
    Set speed to "Middle".

Usage: SpeedHigh
Target: Controller
    Set speed to "High".

Usage: GetSpeedList
Target: Controller, Motor
    Get list of settable motor speed.

Usage: GetAccRateList
Target: Controller, Motor
    Get list of settable motor acceleration rate.

Usage: GetMotorList
Target: Controller
    List motor names.

Usage: GetMotorName MotorNumber
Target: Controller

Get motor name of "MotorNumber".

Usage: GetMotorNumber MotorName
Target: Controller
Get motor number of "MotorName".

#Set Commandst
Usage: Preset MotorNumber Value
Target: Controller
Set motor position data of "MotorNumber" (0 to 16) into "Value".

Usage: Preset Value
Target: Motor
Set motor position data into "Value".

Usage: Select Channel MotorNumber
Target: Controller
Select "MotorName" or "MotorNumber" (0 to 16) on "Channel" (A, B, C, D).

Usage: SetAccRate MotorNumber Value
Target: Controller
Set acceleration rate of "MotorNumber" (0 to 16) into "Value".

Usage: SetAccRate Value
Target: Motor
Set acceleration rate into "Value".

Usage: SetCancelBacklash MotorNumber Value
Target: Controller
Set cancel backlash value of "MotorNumber" (0 to 16) into "Value".

Usage: SetCancelBacklash Value
Target: Motor
Set cancel backlash value into "Value".

Usage: SetDigitalCcwLs MotorNumber Value
Target: Controller
Set CCW software limit switch (DIGITAL LS) of "MotorNumber" (0 to 16) into "Value".

Usage: SetDigitalCcwLs Value
Target: Motor
    Set CCW software limit switch (DIGITAL LS) into "Value".

Usage: SetDigitalCwLs MotorNumber Value
Target: Controller
    Set CW software limit switch (DIGITAL LS) of "MotorNumber" (0 to 16) into "Value".

Usage: SetDigitalCwLs Value
Target: Motor
    Set CW software limit switch (DIGITAL LS) into "Value".

Usage: SetFunction 1|0
Target: Controller
    Set function (Remote=1, Local=0).

Usage: SetHighSpeed MotorNumber Value
Target: Controller
    Set high speed of "MotorNumber" (0 to 16) into "Value".

Usage: SetHighSpeed Value
Target: Motor
    Set high speed into "Value".

Usage: SetHold Channel 1|0
Target: Controller
    Set hold (=1) or free (=0) motor on "Channel" (A, B, C, D).

Usage: SetJogPulse MotorNumber Value
Target: Controller
    Set jog pulse value of "MotorNumber" (0 to 16) into "Value".

Usage: SetJogPulse Value
Target: Motor
    Set jog pulse value into "Value".

Usage: SetLimits MotorNumber Value
Target: Controller
    Set limit switch value of "MotorNumber" (0 to 16) into "Value".

bit 0: CW LS A/B, 1: CCW LS A/B, 2: Z LS A/B, 3: CW LS ENABLE
4: CCW LS ENABLE, 5: DIGITAL LS ENABLE, 6: HOLD, 7: MOTOR OFF

Usage: SetLimits
Target: Motor
Set limit switch value into "Value".
bit 0: CW LS A/B, 1: CCW LS A/B, 2: Z LS A/B, 3: CW LS ENABLE
4: CCW LS ENABLE, 5: DIGITAL LS ENABLE, 6: HOLD, 7: MOTOR OFF

Usage: SetLowSpeed MotorNumber Value
Target: Controller
Set low speed of "MotorNumber" (0 to 16) into "Value".

Usage: SetLowSpeed Value
Target: Motor
Set low speed into "Value".

Usage: SetMiddleSpeed MotorNumber Value
Target: Controller
Set middle speed of "MotorNumber" (0 to 16) into "Value".

Usage: SetMiddleSpeed Value
Target: Motor
Set middle speed into "Value".

#Move and stop
Usage: SetValue Channel|MotorNumber Value
Target: Controller
Move motor which is shown "MotorNumber" or "Channel" to "Value" absolutely.

Usage: SetValue Value
Target: Motor
Move motor to "Value" absolutely.

Usage: SetValueREL Channel|MotorNumber Value
Target: Controller
Move motor which is shown "MotorNumber" or "Channel" to "Value" relatively.

Usage: SetValueREL Value
Target: Motor

Move motor to "Value" relatively.

Usage: JogCw Channel|MotorNumber
Target: Controller
Send CW jog command to "Channel" (A, B, C, D) or "MotorNumber" (0 to 16).

Usage: JogCw
Target: Motor
Send CW jog command.

Usage: JogCcw Channel|MotorNumber
Target: Controller
Send CCW jog command to "Channel" (A, B, C, D) or "MotorNumber" (0 to 16).

Usage: JogCcw
Target: Motor
Send CCW jog command.

Usage: ScanCw Channel|MotorNumber
Target: Controller
Move "Channel" (A, B, C, D) or "MotorNumber" (0 to 16) to "CW" with scan mode.

Usage: ScanCw
Target: Motor
Move "CW" direction with scan mode.

Usage: ScanCcw Channel|MotorNumber
Target: Controller
Move "Channel" (A, B, C, D) or "MotorNumber" (0 to 16) to "CCW" with scan mode.

Usage: ScanCcw
Target: Motor
Move "CCW" direction with scan mode.

Usage: ScanCwConst Channel|MotorNumber
Target: Controller
Move "Channel" (A, B, C, D) or "MotorNumber" (0 to 16) to "CW" with constant scan mode.

Usage: ScanCwConst

Target: Motor

    Move "CW" direction with constant scan mode.

Usage: ScanCcwConst Channel|MotorNumber

Target: Controller

    Move "Channel" (A, B, C, D) or "MotorNumber" (0 to 16) to "CCW" with constant scan mode.

Usage: ScanCcwConst

Target: Motor

    Move "CCW" direction with constant scan mode.

Usage: ScanCwHome Channel|MotorNumber

Target: Controller

    Move "Channel" (A, B, C, D) or "MotorNumber" (0 to 16) to "CW" for finding home position.

Usage: ScanCwHome

Target: Motor

    Move "CW" direction for finding home position.

Usage: ScanCcwHome Channel|MotorNumber

Target: Controller

    Move "Channel" (A, B, C, D) or "MotorNumber" (0 to 16) to "CCW" for finding home position.

Usage: ScanCcwHome

Target: Motor

    Move "CCW" direction for finding home position.

Usage: Stop [Channel|MotorNumber]

Target: Controller

    Stop motors(s) which shown "Channel" or "MotorNumber". If they are not specified, all motors will be stopped.

Usage: Stop

Target: Motor

    Stop motor.

Usage: StopEmergency [Channel|MotorNumber]

Target: Controller

Make a sudden stop which shown "Channel" or "MotorNumber". If they are not specified, all motors will be stopped.

Usage: StopEmergency

Target: Motor

Make a sudden stop.

#Events

Event: _ChangedIsBusy Value

_ChangedIsBusy event shows that the status of motor has been changed.
1 is busy, 0 is free.

Event: _ChangedCtlIsBusy Value

_ChangedIsBusy event shows that the status of motor has been changed.
1 is busy, 0 is free.

Event: _ChangedValue Value

_ChangedValue event shows that the position of motor has benn changed.

Event: _ChangedFunction Value

_ChangedFunction event event shows that the function has been changed.
0 is Local, 1 is Remote.

4.2　STARStoWE7000 用コマンド概要

2005 年 10 月版

STARS 経由で WE7000 の各モジュールを使用するには、下記フォーマットのメッセージを送信して行います。

メッセージ配信先名□コマンド□引数(必要な場合のみ)
※ □は半角スペースを意味しています

例）　WE7000.mca01 Run

メッセージを送った場合は必ず返事（リプライメッセージを含んだ文字列）が返ってきます。なお、用意されていないコマンドを含んだメッセージを送った場合は下記のエラーを含んだ文字列が返ってきます。

(メッセージ送信元)>(メッセージ配信先)□Bad Command

なお、現時点で対応しているモジュールは下記の通りです。

＜WE7562 (MCA モジュール)＞

[メッセージ配信先名]

WE7000.mca01

Input1 **に対して命令（コマンド）を送信する場合の配信先名**

WE7000.mca02

Input2 **に対して命令（コマンド）を送信する場合の配信先名**

[コマンド]

Stop

このコマンドを送信するとモジュールの動作を終了します。

[リプライ・メッセージ]

@Stop Ok:	正常に動作を終了した場合
@Stop Er: (エラーコード)	エラーで正常終了できなかった場合 （エラーコードは数字の文字列です。エラーコードの詳細は WE コントロール API のマニュアルを参照してください。）

[例]

（送信側）

WE7000.mca01 Stop	Input1 の動作を終了させる場合

（返信されてくる文字列）

WE7000.mca01>TEST @Stop Er:80	80 のエラーコードが原因で終了しなかった場合
WE7000.mca01>TEST @Stop Ok:	正常に終了した場合

※ “TEST”と名づけられたクライアントでの受け取り例（以下全て同様）

Run

このコマンドを送信するとモジュールが動作を開始します。すでに動作中の場合はモジュールに対しての動作指令を行わず、その旨のリプライ・メッセージが帰ってきます。

[リプライ・メッセージのコマンド]

@Run Ok :	正常に動作が開始された場合
@Run Ng: Running	既に動作状態である場合
@Run Er: (エラーコード)	エラーで動作が開始されなかった場合 （エラーコードは数字の文字列です。エラーコードの詳細は WE コントロール API のマニュアルを参照してください。）

[例]

（送信側）

WE7000.mca02 Run	Input2 の動作を開始する場合

（返信されてくる文字列）

WE7000.mca02>TEST @Run Er: 205	205 のエラーコードが原因で動作しなかった場合
WE7000.mca02>TEST @Run Ok:	正常に動作開始した場合

GetValue

このコマンドを送信することで MCA モジュール各 Input の最新データ取得を行います。
このコマンドはいつでも送信可能です。

[リプライ・メッセージのコマンド]

@GetValue データ	データはカンマ区切りの Strings 型です。データの量は分解能（チャネル数）の設定によって変化します。（デフォルトでは 4096 個のデータ）

[例]

（送信側）

WE7000.mca01 GetValue　　Input1 のデータを取得する場合

（返信されてくる文字列）

WE7000.mca01>TEST @GetValue 0,0,0,12,25,45,16,4,……

IsBusy

このコマンドを送信することで各 Input の動作状況を知ることが可能です。
このコマンドはいつでも送信可能です。

[リプライ・メッセージ]

@IsBusy 1	対象 Input が動作中の場合
@IsBusy 0 `	対象 Input が停止中の場合

[例]

（送信側）

WE7000.mca02 IsBusy　　Input2 の動作状況を確認する場合

（返信されてくる文字列）

WE7000.mca02>TEST @IsBusy 0	Inpu2 が停止中の場合
WE7000.mca02>TEST @IsBusy 1	Inpu2 が動作中の場合

SetMode

このコマンドを送信することで、MCA **モジュールの動作モード切替を行います。**

このコマンドを動作中に送るとエラーになります。

MCS **から** PHA **モードに切り替えた場合は分解能がデフォルト値**(4096)**に設定されます。**

[**引数**]

PHA	PHA **モードに変換します**
MCS	MCS **モードに変換します**

[**リプライ・メッセージのコマンド**]

@SetMode PHA Ok:	**正常に** PHA **モードに切り替わった場合**
@SetMode MCS Ok:	**正常に** MCS **モードに切り替わった場合**
@SetMode PHA Ng: Now, PHA mode”	PHA **モードへの切替コマンドを送ったが既に** PHA **モードだった場合**
@SetMode MCS Ng: Now, MCS mode”	MCS **モードへの切替コマンドを送ったが既に** MCS **モードだった場合**
@SetMode PHA Er: (**エラーコード**)	PHA **モードへの切替コマンドを送ったがエラーで切替が行われなかった場合** **（エラーコードは数字の文字列です。エラーコードの詳細は** WE **コントロール** API **のマニュアルを参照してください）**
@SetMode MCS Er: (**エラーコード**)	MCS **モードへの切替コマンドを送ったがエラーで切替が行われなかった場合** **（エラーコードは数字の文字列です。エラーコードの詳細は** WE **コントロール** API **のマニュアルを参照してください）**

[**例**]

（送信側）

WE7000.mca02 SetMode PHA　　Input2 **を** PHA **モードに変更する場合**

（返信されてくる文字列）

WE7000.mca02>TEST @SetMode PHA Ok:

Inpu2 **のモードが正常に** PHA **モードに切り替わった場合**

WE7000.mca02>TEST @SetMode PHA Ng: Running

Inpu2 **が動作中だった場合**

WE7000.mca02>TEST @SetMode PHA Ng: Now, PHA mode

Inpu2 **が既に** PHA **モードだった場合**

SetChannel

このコマンドを送信することで、PHA モード時の分解能（チャネル数）を指定できます。
MCS モード時や動作中にこのコマンドを送るとエラーになります。
デフォルト値は 4096 に設定されています。また分解能（チャネル数）の切替を行った場合はモジュール内のデータが全てリセットされます。SetMode を使用してｃ

[引数]

128	分解能（チャネル数）を 128 にします
256	分解能（チャネル数）を 256 にします
512	分解能（チャネル数）を 512 にします
1024	分解能（チャネル数）を 1024 にします
2048	分解能（チャネル数）を 2048 にします
4096	分解能（チャネル数）を 4096 にします

[リプライ・メッセージのコマンド]

@SetChannel (引数) Ok:	正常に分解能（チャネル数）の切替が行われた場合
@SetChannel (引数) Er: (エラーコード)	エラーで切替が行われなかった場合（エラーコードは数字の文字列です。エラーコードの詳細は WE コントロール API のマニュアルを参照してください）

[例]

（送信側）

WE7000.mca02 SetChannel 1024	Input2 の分解能（チャネル数）を 1024 に変更する場合

（返信されてくる文字列）

WE7000.mca02>TEST @SetChannel 1024 Ok:
Inpu2 の分解能（チャネル数）が正常に 1024 に切り替わった場合

WE7000.mca02>TEST @SetChannel 1024 Ng: Running
Inpu2 が動作中だった場合

WE7000.mca02>TEST @SetChannel 1024 Ng: Now, MCS mode
Inpu2 が MCS モードだった場合

WE7000.mca02>TEST @SetChannel 1024 Er: 502
502 のエラーコードが原因で分解能（チャネル数）の切替が行われなかった場合

GetMode

このコマンドを送信することで、MCA モジュールの動作モードの問合わせが行えます。

[リプライ・メッセージのコマンド]

@GetMode Pha	PHA モードの場合
@GetMode Mcs	MCS モードの場合
@Getmode Er: (エラーコード)	エラーで問合わせが行われなかった場合（エラーコードは数字の文字列です。エラーコードの詳細は WE コントロール API のマニュアルを参照してください）

[例]

（送信側）

WE7000.mca01 GetMode	Input1 の動作モードを問合わせる場合

（返信されてくる文字列）

WE7000.mca01>TEST @GetMode Pha	Inpu1 の動作モードが PHA モードの場合
WE7000.mca01>TEST @GetMode Mcs	Inpu1 の動作モードが MCS モードの場合
WE7000.mca01>TEST @GetMode Er: 501	501 のエラーコードで問合わせが行えなかった場合

GetChannel

このコマンドを送信することで、分解能（チャネル数）の問合わせが行えます。

[リプライ・メッセージのコマンド]

@GetChannel (チャネル数)	現在のチャネル数が返ってきます。
@Getmode Er: (エラーコード)	エラーで問合わせが行われなかった場合（エラーコードは数字の文字列です。エラーコードの詳細は WE コントロール API のマニュアルを参照してください）

[例]

（送信側）

WE7000.mca02 GetChannel	Input1 の分解能（チャネル数）を問合わせる場合

（返信されてくる文字列）

WE7000.mca02>TEST @GetChannel 1024	Inpu1 の分解能が 1024 の場合
WE7000.mca02>TEST @GetMode Er: 501	501 のエラーコードで問合わせが行えなかった場合

＜WE7521 (4ch タイミングカウンターモジュール)＞

[**メッセージ配信先名**]

WE7000.counter01

CH1 **に対して命令（コマンド）を送信する場合の配信先名**

WE7000.counter02

CH2 **に対して命令（コマンド）を送信する場合の配信先名**

WE7000.counter03

CH1 **に対して命令（コマンド）を送信する場合の配信先名**

WE7000.counter04

CH2 **に対して命令（コマンド）を送信する場合の配信先名**

WE7000.counter05

CH1 **に対して命令（コマンド）を送信する場合の配信先名**

WE7000.counter06

CH2 **に対して命令（コマンド）を送信する場合の配信先名**

WE7000.counter07

CH1 **に対して命令（コマンド）を送信する場合の配信先名**

WE7000.counter08

CH2 **に対して命令（コマンド）を送信する場合の配信先名**

WE7000.counter00

全チャンネルに対して命令（コマンド）を送信する場合の配信先名

[コマンド]

Stop

このコマンドを送信するとモジュールの動作を終了します。

[リプライ・メッセージ]

@Stop Ok: (測定値)	正常に動作を終了した場合。終了した時点での測定値が引数として返ってきます（メッセージ配信先に counter00 を指定した場合は、カンマ区切りで 8 個分 (モジュール 2 枚分) のデータが返ってきます）
@Stop Er: (エラーコード)	エラーで正常終了できなかった場合（エラーコードは数字の文字列です。エラーコードの詳細は WE コントロール API のマニュアルを参照してください。）

[例]

（送信側）

WE7000.counter01 Stop　　CH1 の動作を終了させる場合

（返信されてくる文字列）

WE7000.counter01>TEST @Stop Er:8082

8082 のエラーコードが原因で終了しなかった場合

WE7000.counter01>TEST @Stop Ok: 230

正常に終了した場合

※ “TEST”と名づけられたクライアントでの受け取り例（以下全て同様）

Run

このコマンドを送信するとモジュールが動作を開始します。すでに動作中の場合はモジュールに対しての動作指令を行わず、その旨のリプライ・メッセージが帰ってきます。

[リプライ・メッセージのコマンド]

@Run Ok :	正常に動作が開始された場合
@Run Ng: Running	既に動作状態である場合
@Run Er: (エラーコード)	エラーで動作が開始されなかった場合（エラーコードは数字の文字列です。エラーコードの詳細は WE コントロール API のマニュアルを参照してください）

[例]

（送信側）

WE7000.counter00 Run	全チャンネルの動作を開始する場合

（返信されてくる文字列）

WE7000.counter00>TEST @Run Er: 8089	8089 のエラーコードが原因で動作しなかった場合
WE7000.counter00>TEST @Run Ok:	正常に動作開始した場合

IsBusy

このコマンドを送信することで各チャンネルの動作状況を知ることが可能です。
このコマンドはいつでも送信可能です。

[リプライ・メッセージ]

@IsBusy 1	対象チャンネルが動作中の場合（メッセージ配信先に counter00 を指定した場合は、全 8 チャンネル中 1 つでも動作中の場合は 1 が返ってきます）
@IsBusy 0 `	対象チャンネルが停止中の場合

[例]

（送信側）

WE7000.counter02 IsBusy	CH2 の動作状況を確認する場合

（返信されてくる文字列）

WE7000.counter02>TEST @IsBusy 0	CH2 が停止中の場合
WE7000.counter02>TEST @IsBusy 1	CH2 が動作中の場合

CounterReset

このコマンドを送信すると指定したチャンネルのデータをリセットして 0 にすることが出来ます。動作中でもデータのリセットを行うことが可能です。この場合はコマンドを受け取った時点でデータがリセットされ、0 からカウントを開始します。

[リプライ・メッセージのコマンド]

@CounterReset Ok　:　　正常にリセットした場合

@Run Er: (エラーコード)　　エラーで動作が開始されなかった場合（エラーコードは数字の文字列です。エラーコードの詳細は WE コントロール API のマニュアルを参照してください）

[例]

（送信側）

WE7000.counter02 CounterReset　　CH2 をリセットして値を 0 にする場合

（返信されてくる文字列）

WE7000.counter02>TEST @CounterReset Er: 8089

8089 のエラーコードが原因でリセット出来なかった場合

WE7000.counter02>TEST @CounterReset Ok:

正常にリセットした場合

GetValue

このコマンドを送信することでデータの取得が可能です。動作していないチャンネルについては、Stop した時のデータが返ってきます。このコマンドはいつでも送信可能です。

[リプライ・メッセージのコマンド]

@GetValue (データ)　　データは Strings 型の数値です。メッセージ配信先に counter00 を指定した場合は、カンマ区切りの 8 個のデータが返ってきます。

[例]

（送信側）

WE7000.counter03 GetValue　　CH3 のデータを取得する場合

（返信されてくる文字列）

WE7000.counter0301>TEST @GetValue 340

SetLevel

このコマンドを送信すると指定したチャンネルの入力しきい値を変更することが出来ます。デフォルトは0になっています。

[引数]

-20.0～20.0 の間のテキスト	V 単位の数値（テキスト）で 0.1V ステップで設定可能

[リプライ・メッセージのコマンド]

@SetLevel (設定した値) Ok :	正常にしきい値が設定された場合
@SetLevel Er: (エラーコード)	エラーでしきい値の設定が出来なかった場合（エラーコードは数字の文字列です。エラーコードの詳細は WE コントロール API のマニュアルを参照してください）

[例]

（送信側）

WE7000.counter07 SetLevel 5.0　　CH7 のしきい値を 5.0V にする場合

（返信されてくる文字列）

WE7000.counter07>TEST @SetLevel Er: 205

205 のエラーコードが原因でしきい値の設定が出来なかった場合

WE7000.counter07>TEST @SetLevel 5.0 Ok:

正常にしきい値の設定が行われた場合

＜WE7262 (32bit デジタル I/O モジュール)＞

[メッセージ配信先名]

WE7000.encorder01

上記のみです。

[コマンド]

SetIOSelect

このコマンドを送信して各 bit の入出力を設定します。

[引数]

0～4294967295(0xffffffff) 出力を行う bit は 1 にして、入力を行う bit は 0 に設定して 32bit 分を設定してください

[リプライ・メッセージのコマンド]

@SetIOSelect (引数) Ok : 正常に設定された場合

@SetIOSelect (引数) Er: (エラーコード)

エラーで設定が出来なかった場合
（エラーコードは数字の文字列です。エラーコードの詳細は WE コントロール API のマニュアルを参照してください）

[例]

（送信側）

WE7000.encoder01 SetIOSelect 4294901760

16bit 目～31bit 目(Input1)までを出力(Onput), 0bit 目～15bit 目(Input2)を入力(Input)に設定する場合

（返信されてくる文字列）

WE7000.counter07>TEST @SetIOSelectl 4294901760 Er: 205

205 のエラーコードが原因で設定が出来なかった場合

WE7000.counter07>TEST @SetIOSelect 4294901760 Ok:

正常に設定が行われた場合

GetIOSelect

このコマンドを送信して各 bit の入出力設定の問い合わせを行います。

[リプライ・メッセージのコマンド]

@GetIOSelect Ok : (戻り値)	問い合わせが正常に終了した場合
@GetIOSelect Er: (エラーコード)	エラーで問い合わせが出来なかった場合（エラーコードは数字の文字列です。エラーコードの詳細は WE コントロール API のマニュアルを参照してください）

[例]

（送信側）

WE7000.encoder01 GetIOSelect	問い合わせを行う場合

（返信されてくる文字列）

WE7000.counter07>TEST @GetIOSelectl Er: 205

205 のエラーコードが原因で問い合わせが出来なかった場合

WE7000.counter07>TEST @GetIOSelect Ok: 4294901760

正常に問い合わせが行われた場合（この場合の戻り値は 0bit 目〜15bit 目(Input2)が入力(Input)で 16bit 目〜31bit 目(Input1)が出力(Output)の設定になっていることを表しています）

GetValue

このコマンドを送信して各bit**のデータ取得が行えます。**

[**リプライ・メッセージのコマンド**]

@GetValue Ok: (**データ**)	**データ取得が正常に行われた場合** **（返ってくるデータは各ビットのデータを** 10 **進数に変更した値になります。各** bit **は、入力の場合** 1 **が入力ありを表しています。出力の場合** 1 **が出力中であることを表しています）**
@GetValue Er: (**エラーコード**)	**エラーでデータ取得が出来なかった場合** **（エラーコードは数字の文字列です。エラーコードの詳細は** WE **コントロール** API **のマニュアルを参照してください）**

[**例**]

（送信側）

WE7000.encoder01 GetValue　　**データ取得を行う場合**

（返信されてくる文字列）

WE7000.counter07>TEST @GetValue Er: 205

205 **のエラーコードが原因でデータ取得が出来なかった場合**

WE7000.counter07>TEST @GetValue Ok: 4294901760

正常に問い合わせが行われた場合

（この場合のデータは全ての bit **が入力に設定してあるときは、**0bit **目〜**15bit **目**(Input2)**が入力ありで** 16bit **目〜**31bit **目**(Input1)**が入力なしを表しています）**

SetValue

このコマンドを送信して各 bit のデータ取得が行えます。

[引数]

0〜4294967295(0xffffffff)　　出力を行う bit を 1 にして 32bit 分を設定してください
（入力に設定した bit に対して 1 を設定しても何も起こらずに 0 に戻ります）

[リプライ・メッセージのコマンド]

@SetValue (引数) Ok: (データ)　　設定が正常に行われた場合

@SetValue (引数) Er: (エラーコード)　　エラーで設定が出来なかった場合
（エラーコードは数字の文字列です。エラーコードの詳細は WE コントロール API のマニュアルを参照してください）

[例]

（送信側）

WE7000.encoder01 SetValue 2　　全 bit 全て出力の設定だとして 2bit 目だけを出力する場合

（返信されてくる文字列）

WE7000.counter07>TEST @SetValue 2 Er: 205

205 のエラーコードが原因で設定が出来なかった場合

WE7000.counter07>TEST @SetValue 2 Ok: 4294901760

正常に設定が行われた場合

4.3　STARS sc400 用コマンド集

2005.12.01 版

STARS 経由でパルスモータ sc400（神津精機製）を使用するには、下記フォーマットのメッセージを送信して行います。

メッセージ配信先名?コマンド?引数(必要な場合のみ)
※ ? は半角スペースを意味しています

例） sc400.theta?GetValue?1

メッセージを送った場合は必ず返事（リプライメッセージを含んだ文字列）が返ってきます。

(メッセージ配信先)>(メッセージ送信元)?@コマンド?引数?値

例） sc400.theta>test?@GetValue?1?10000

対応しているメッセージ配信先名は下記の通りです。

[**メッセージ配信先名**]
Stars のノード名が sc400 の場合（Stars のノード名は sc400 I/O Client プログラム起動に指定します）

sc400	**コントローラコマンド** **パルスモータのコントローラに対してメッセージを配信します**
sc400.*motorname*	**モータコマンド** motorname **という名前のモータに対してメッセージを配信します**
sc400.*encodername*	**エンコーダタコマンド** encodername **という名前のエンコーダに対してメッセージを配信します**

エラーメッセージについて

エラーが起こると返事（リプライメッセージ）として下記の形式の文字列が返ってきます。

(メッセージ配信先)>(メッセージ送信元)?@コマンド?引数?Er:?（エラー内容を表す文字列）

例）　sc400.theta>test?@Preset?1000000000?Er:?Preset?Out?Of?Range.

メッセージ配信先を誤って送った場合は下記のエラーを含んだ文字列が返ってきます。

(sc400 のノード名)>(メッセージ送信元)?@コマンド?引数?Er:?（誤って送ったメッセージ配信先名）?is?down.

例）　sc400.thet?GetValue?1
sc400>test?@GetValue?1?Er:?sc400.thet?is?down.

用意されていないコマンドもしくは適切でない引数を含んだメッセージを送った場合は下記のエラーを含んだ文字列が返ってきます。

(メッセージ配信先)>(メッセージ送信元)?@コマンド?引数?Er:?Bad?command?or?parameter

例）　sc400.theta?GetValued?1
sc400>test?@GetValued?1?Er:?Bad?command?or?parameter

SC 本体にコマンドを送信してエラーで返ってきた場合は下記のエラーを含んだ文字列が返ってきます。

(メッセージ配信先)>(メッセージ送信元)?@コマンド?引数?Er:?E?（エラーコード※）

※エラーコードは数字の文字列です。
エラーコードの詳細は SC 本体の取扱説明書のエラーコード一覧を参照してください。

例）　sc400.theta?SetValue?1?2?0?0?0
sc400>test?@?SetValue?1?2?0?0?0Er:?E?302

プログラムとして想定外のエラーが発生した場合は下記のエラーを含んだ文字列が返ってきます。
この場合、お手数ですがプログラム担当者までご連絡ください。

(メッセージ配信先)>(メッセージ送信元)?@コマンド?引数?Er:?SYS?（メッセージ文字列）

コントローラコマンド

[メッセージ配信先名]

sc400

コントローラに対して命令（コマンド）を送信する場合の配信先名

[コマンド]

hello

STARS の通信が行われているかをチェックするコマンド。

このコマンドを送信すると'@hello nice to meet you.'の文字列を返します。

[例]

（送信側）

sc400?hello

（返信されてくる文字列）

sc400>test?@hello?nice?to?meet?you.

GetMotorList

このコマンドを送信するとモータ名称の一覧をスペース区切りで返します。

[例]

（送信側）

sc400?GetMotorList

（返信されてくる文字列）

sc400>test?@GetMotorList?theta?DTH?D1?PH2　　左から順に軸 No.1 の名称'theta'、軸 No.2 の名称'DTH'、軸 No.3 の名称'D1'、軸 No.4 の名称'PH2'を返します

モータの名称の定義方法につきましては、sc400 取扱説明書を参照してください。

GetEncoderList

このコマンドを送信するとエンコーダ名称の一覧をスペース区切りで返します。

[例]

（送信側）

sc400?GetEncoderList

（返信されてくる文字列）

sc400>test?@GetEncoderList?ENCtheta?ENCDTH?ENCD1?ENCPH2

左から順に軸 No.1 の名称 'ENCtheta'、軸 No.2 の名称

'ENCDTH'、軸 No.3 の名称 'ENCD1'、軸 No.4 の名称'ENCPH2' を返します

エンコーダの名称の定義方法につきましては、sc400 取扱説明書を参照してください。

GetMotorName

軸 No.に対応するモータ名を返します。

[引数]

軸 No.　　1 からコントローラの軸の数までの数値文字

[リプライ・メッセージ]

@GetMotorName?（引数）?（モータ名）	データ取得が正常におこなわれた場合
@GetMotorName?（引数）?Er:?Bad?command?or?parameter	軸 No.が間違っていてエラーの場合

[例]

（送信側）

sc400?GetMotorName 1	軸 No.1 のモータ名を問い合わせます

（返信されてくる文字列）

sc400>test?@GetMotorName?1?theta	軸 No.1 のモータ名'theta'が返ってきます

GetEncoderName

軸 No.に対応するエンコーダ名を返します。

[引数]

軸 No.　　1 からコントローラの軸の数までの数値文字

[リプライ・メッセージ]

@GetEncoderName?（引数）?（モータ名）	データ取得が正常におこなわれた場合
@GetEncoderName?（引数）? Er:?Bad?command?or?parameter	軸 No.が間違っていてエラーの場合

[例]

（送信側）

sc400?GetEncoderName 1	軸 No.1 のエンコーダ名を問い合わせます

（返信されてくる文字列）

sc400>test?@GetEncoderName?1?ENCtheta	軸 No.1 のエンコーダ名'ENCtheta'が返ってきます

flushdata

このコマンドを送信するとコントローラおよびモータ、エンコーダの全てのステータス情報をイベントメッセージとして Stars の TAK サーバ'System'に返します。
ステータス情報をイベントメッセージとして受け取るには、この当コマンドを発行する前に Stars の TAK サーバ'System' に対してイベントメッセージ配信依頼のコマンドを送信しておく必要があります。

[リプライ・メッセージ]

@flushdata?Ok:	コマンドが正常に送信された場合

[例]

（送信側）

System?flgon?sc400.theta	モータ名'theta'のイベントメッセージの配信を依頼します
System?flgon?sc400.ENCtheta	エンコーダ名'ENCtheta'のイベントメッセージの配信を依頼します
sc400?flushdata	イベントメッセージ配信の実行を依頼します

（返信されてくる文字列）

sc400?@flushdata?Ok:	コマンドが正常送信されました
sc400.theta>test?_ChangedIsBusy?1	モータ名 theta の Busy 状態がイベントメッセージの値として返ってきます
sc400.theta>test?_ChangedValue?100	モータ名 theta の現在値がイベントメッセージの値として返ってきます
sc400.ENCtheta>test?_ChangedValue?101	エンコーダ名 ENCtheta の現在値がイベントメッセージの値として返ってきます

※ _ChangedValue イベントの返す現在位置の値については、あらかじめ設定ファイル config.pl に現在位置としてどのような形式の値を返すのかパラメータとして記述しておく必要があります。詳細は SC400 取扱説明書を参照してください。

Stop

このコマンドを送信すると全てのモータが減速停止します。

[リプライ・メッセージ]

@Stop?Ok:	正常に動作を終了した場合
@Stop?Er:?E?(エラーコード)	SC 本体のエラーで正常終了できなかった場合

[例]

（送信側）

sc400?Stop	全てのモータを減速停止します

（返信されてくる文字列）

sc400>test?@Stop?@Ok:　正常に動作を終了した場合
sc400>test?@Stop?Er:?E?(エラーコード)　SC 本体のエラーで正常終了できなかった場合

StopEmergency

このコマンドを送信すると全てのモータが緊急停止します。

[リプライ・メッセージ]

@StopEmergency?Ok:　正常に動作を終了した場合
@StopEmergency?Er:?E?(エラーコード)　SC 本体のエラーで正常終了できなかった場合

[例]

（送信側）

sc400?StopEmergency　全てのモータを緊急停止します

（返信されてくる文字列）

sc400>test?@StopEmergency?@Ok:　正常に動作を終了した場合
sc400>test?@StopEmergency?Er:?E?(エラーコード)　SC 本体のエラーで正常終了できなかった場合

Standby

２軸（機種 SC200、SC400、SC800）、３軸あるいは４軸（機種 sc400、SC800）の同時駆動 Standby 状態にします。

[リプライ・メッセージ]

@Standby?Ok:　正常に動作を終了した場合
@Standby?Er:?Standby?On.　既に多軸同時駆動 Standby の状態でエラーの場合

[例]

（送信側）

sc400?Standby　多軸同時駆動 Stanby 状態にします。

（返信されてくる文字列）

sc400>test?@Standby?@Ok:　正常に動作を終了した場合
sc400>test?@Standby?@Er:?Standby?On.　既に多軸同時駆動 Standby の状態でエラーの場合

SyncRun

このコマンドを送信すると多軸同時駆動の動作を開始します。

[リプライ・メッセージ]

@SyncRun?Ok:	正常に動作を終了した場合
@SyncRun?Er:?Standby?Off.	多軸同時駆動 Standby の状態でないためエラーの場合
@SyncRun?Er:?SetValue?Not?Executed.	SetValue コマンドが一回も実行されていないためエラーの場合
@SyncRun?Er:?SetValue?1?Time?Executed.	SetValue コマンドが一回しか実行されていないためエラーの場合
@SyncRun?Er:?E?(エラーコード)	SC 本体のエラーで正常終了できなかった場合

[例]

（送信側）

sc400?SyncRun	多軸同時駆動の動作を開始します

（返信されてくる文字列）

sc400>test?@SyncRun?@Ok:	正常に動作を終了した場合
sc400>test?@SyncRun?Er:?E?(エラーコード)	SC 本体のエラーで正常終了できなかった場合

GetMachineName

このコマンドを送信するとコントローラ本体の機種名を返します。

[例]

（送信側）

sc400?GetMachineName

（返信されてくる文字列）

sc400>test?@GetMachineName?SC-400	機種名 SC-400 が返ってきます

GetVersion

このコマンドを送信するとコントローラのシステムプログラムのバージョンを返します。

[例]

（送信側）

sc400?GetVersion

（返信されてくる文字列）

sc400>test?@GetVersion?0.997	バージョン 0.997 が返ってきます

モータコマンド

[メッセージ配信先名]

sc400.*motorname*

motorname のモータに対して命令（コマンド）を送信する場合の配信先名

GetAxisNumber

このコマンドを送信することでモータ名に対応する軸番号を取得します。

[例]

（送信側）

sc400.theta?GetAxisNumber　　モータ theta の軸番号を取得します

（返信されてくる文字列）

sc400.theta>test?@GetAxisNumb er? 1　　モータ theta の軸番号 1 が返ってきます

GetSysInfo

このコマンドを送信することでモータのシステム情報のデータを取得します。

[引数]

＜パターン１＞

①なし

SC コマンド「RSY」の返答データをシステム No.1 から 47 まで全て,（カンマ）区切りで返します

＜パターン２＞

①システム No.

1 から 47 までの数字文字列

引数のシステム No.に対応する SC コマンド「RSY」の返答データを返します

＜パターン３＞

①(開始システム No.)-（終了システム No.）

- 開始システム No.と終了システム No.を半角ハイフンでつなげます
- 開始システム No.と終了システム No は、1 から 47 までの数字文字列を指定します

開始システム No.から終了システム No.までの SC コマンド「RSY」の返答データを,（カンマ）区切りで返します

[リプライ・メッセージ]

@GetSysInfo?（引数）? システム No.a? 値 a,システム No.b? 値 b,…,システム No.b? 値 z

正常に動作を終了すると{SystemNo.? 値}の組み合わせで、システム No に対応する値が返ってきます

@GetSysInfo?（引数）?Er:?Bad?command?or?parameter

引数の形式を間違えてエラーの場合

@GetSysInfo?（引数）?Er:?E?(エラーコード)　SC 本体のエラーで正常終了できなかった場合

[例１]

（送信側）

sc400.theta?GetSysInfo　モータ theta のシステム情報 No.1～47 のデータを取得

（返信されてくる文字列）

sc400.theta>test?@GetSysInfo?1?500,2?5000,3?24,4?24,5?0,6?0,7?0,8?0,9?3,10?1,11?1,12?2,13?0,14?100,15?0,16?1,17?1,18?0,19?0,20?0,21?0,22?2,23?5,24?1,25?1,26?1,27?0,28?2,29?0,30?1,31?100,32?100,33?0,34?0,35?1,36?0,37?0,38?1,39?0,40?1,41?0,42?0,43?2,44?0,45?0,46?1,47?0　{SystemNo.?値} の組み合わせが「,」(カンマ区切り)で全てのシステム情報 No.１から～No.４７まで返ってきます

sc400.theta>test?@GetSysInfo?Er: E (エラーコード)　SC 本体のエラーで正常終了できなかった場合

[例２]

（送信側）

sc400.theta?GetSysInfo?9　モータ theta のシステム情報 No.9 のデータを取得

（返信されてくる文字列）

sc400.theta>test?@GetSysInfo?9?9?3　モータ theta のシステム情報 No.9 の値 3 を取得

sc400.theta>test?@GetSysInfo?9?Er: E (エラーコード)　SC 本体にのエラーで正常終了できなかった場合

[例３]

（送信側）

sc400.theta?GetSysInfo?9-13　theta のシステム情報 No.9～13 のデータを取得

（返信されてくる文字列）

sc400.theta>test?@GetSysInfo?9-13?9?3,10?1,11?1,12?2,13?0　{SystemNo.?値} の組み合わせが「,」(カンマ区切り)で全てのシステム情報 No.9 から～No.13 まで返ってきます

sc400.theta>test?@GetSysInfo?9-13?Er: E (エラーコード)

SC 本体のエラーで正常終了できなかった場合

GetSpeedTblInfo

このコマンドを送信することで速度テーブルのデータとして SC コマンド「RTB」コマンドの値をスペース区切りで返します。

[引数]

速度テーブル No.　　1〜9 の数字文字

[リプライ・メッセージ]

@GetSpeedTblInfo?（引数）?（設定方法確認）?（スタート速度）?（最高速度）?（加速パルス数）?（減速パルス数）?（加速時間）?（減速時間）

正常に動作を終了した場合

@GetSysInfo?（引数）?Er:?Bad?command?or?parameter　　引数の速度テーブル No が有効範囲を超えたためエラーの場合

@GetSysInfo?Er:?E?(エラーコード)　　SC 本体のエラーで正常終了できなかった場合

[例]

（送信側）

sc400.theta?GetSpeedTblInfo?1

モータ theta の速度テーブル No.1 の設定情報のデータを取得

（返信されてくる文字列）

sc400.theta>test?@GetSpeedTblInfo?1?1?501?2001?3125?3125?250?250

モータ theta の速度テーブル No.1 の設定情報のデータが返ってきます

sc400.theta>test?@SpeedTblInfo?Er:?E?(エラーコード)

SC 本体のエラーで正常終了できなかった場合

SetSpeedTblInfo

このコマンドを送信することで速度テーブルのデータを設定します。

このコマンドはメッセージ配信先として指定したモータが Busy 状態もしくは多軸同時駆動 Standby 状態のときはエラーとなり実行されません。

[引数]

①速度テーブル No.　　1〜9 の数字文字

②スタート速度※

③最高速度※

④加速時間※

⑤加速時間※

※　②～⑤の引数が取りうる値の範囲は SC コマンド「WTB」をご確認ください。
Stars プログラムによる値の範囲チェックはおこなっておりません。

[リプライ・メッセージ]

@SetSpeedTblInfo?（引数）?Ok:	正常に動作を終了した場合
@SetSpeedTblInfo?（引数）?Er:?Bad?command?or?parameter	引数の速度テーブル No が有効範囲を超えたためエラーの場合
@SetSpeedTblInfo（引数）?Er:?Busy.	モータが Busy 状態のため実行されなかった場合
@ SetSpeedTblInfo（引数）?Er:?Standby?On.	多軸同時駆動 Standby 状態のため実行されなかった場合
@SetSpeedTblInfo?（引数）?Er:?E?(エラーコード)	SC 本体のエラーで正常終了できなかった場合

[例]

（送信側）

sc400.theta?SetSpeedTblInfo?1?501?2001?25?25　theta の速度テーブル No.1 の設定情報のデータを更新する

（返信されてくる文字列）

sc400.theta>test?@SetSpeedTblInfo?1?1?501?2001?25?25?Ok:
正常に動作を終了した場合

sc400.theta>test?@SetSpeedTblInfo?1?1?501?2001?25?25?Er:?E?(エラーコード)
SC 本体のエラーで正常終了できなかった場合

SetLink

このコマンドを送信することでメッセージ配信先として指定したモータに従属するモータとモータ電子カップリング比率を指定することができます。
このコマンドはメッセージ配信先として指定したモータが Busy 状態もしくは多軸同時駆動 Standby 状態のときはエラーとなり実行されません。
当コマンドの詳細な仕様に関しては、SC 本体の取扱説明書にあるコマンド「LNK」をご確認ください。

[引数]

①従属モータ名１
メッセージ配信先として指定したモータに従属して動作する１番目のモータ名を指定します

②従属モータ１の比率
1 から 256 までの数字文字列

③従属モータ名２

オプションで省略可能です

メッセージ配信先として指定したモータに従属して動作する２番目のモータ名を指定します

④従属モータ２の比率

③の従属モータ名２を指定した場合は必須です

1 から 256 までの数字文字列

[リプライ・メッセージ]

@SetLink?（引数）?Ok:	正常に動作を終了した場合
@SetLink?（引数）?Er:?Bad?Motor?Name	モータ名に誤りがあってエラーの場合
@SetLink?（引数）?Er:?Bad?command?or?parameter	従属モータの比率の値が間違っていてエラーの場合
@SetLink?（引数）?Er:?Busy.	モータが Busy 状態のため実行されなかった場合
@SetLink?（引数）?Er:?Standby?On.	多軸同時駆動 Standby 状態のため実行されなかった場合
@SetLink?（引数）?Er:?E?(エラーコード)	SC 本体のエラーで正常終了できなかった場合

[例　メッセージ配信先として指定したモータに従属するモータが１つの場合]

（送信側）

sc400.theta?SetLink?DTH?2	モータ theta に対してモータ DTH を電子カップリング比率 2 で従属させます

（返信されてくる文字列）

sc400.theta>test?@SetLink?DTH?2?Ok:	正常に動作を終了した場合
sc400.theta>test?@SetLink?DTH?2?Er:?(エラーコード)	SC 本体のエラーで正常終了できなかった場合

[例　メッセージ配信先として指定したモータに従属するモータが２つの場合]

（送信側）

sc400.theta?SetLink?DTH?2?D1?3	モータ theta に対してモータ DTH を電子カップリング比率 2、モータ D1 を電子カップリング比率 3 で従属させます

（返信されてくる文字列）

sc400.theta>test?@SetLink?DTH?2?D1?3?Ok:	正常に動作を終了した場合
sc400.theta>test?@SetLink?DTH?2?D1?3?Er:?(エラーコード)	SC本体のエラーで正常終了できなかった場合

ScanHome

このコマンドを送信することでモータの原点位置検出をおこないます。

このコマンドはメッセージ配信先として指定したモータが Busy 状態もしくは多軸同時駆動 Standby 状態のときはエラーとなり実行されません。

[引数]

①原点復帰モード

1 から 14 までの数値文字列

②加減速モード

1 から 5 までの数字文字

③同期モード

0 もしくは 1 の数字文字

④速度テーブル No.

0 から 9 までの数字文字

※①～④の引数について詳細を知りたい場合は、SC 本体の取扱説明書にあるコマンド「ORG」をご確認ください。

[リプライ・メッセージ]

@ScanHome?（引数）?Ok:	正常に動作を終了した場合
@ScanHome?（引数）?Er:?Bad?command?or?parameter	引数①から④の値が有効範囲を超えたためエラー場合
@ScanHome?（引数）?Er:?Busy.	モータがBusy 状態のため実行できなかった場合
@ScanHome?（引数）?Er:?Standby?On.	多軸同時駆動 Standby 状態のため実行されなかった場合
@ScanHome?（引数）?Er:? E?(エラーコード)	SC 本体のエラーで正常終了できなかった場合

[例]

（送信側）

sc400.theta?ScanHome?2?0?1?0?3　　モータ theta の原点位置検出を方式 3 でおこないます

（返信されてくる文字列）

sc400.theta>test?@ScanHome?2?0?1?0?3?Ok:　　正常に動作を終了した場合

sc400.theta>test?@ScanHome?2?0?1?0?3?Er:?E?(エラーコード)

SC 本体のエラーで正常終了できなかった場合

Preset

このコマンドを送信することでモータの現在位置を指定値で書き換えます。
このコマンドはメッセージ配信先として指定したモータが Busy 状態もしくは多軸同時駆動 Standby 状態のときはエラーとなり実行されません。

[引数]

モータの値

-68108813 から～68108813 までの数値文字列（＋符号の指定は不可）

[リプライ・メッセージ]

@Preset?（引数）?Ok:	正常に動作を終了した場合
@Preset?（引数）?Er:? Preset?Out?Of?Range.	引数のモータの値が有効範囲を超えたためエラーの場合
@Preset?（引数）?Er:?Busy.	モータが Busy 状態のため実行されなかった場合
@Preset?（引数）?Er:?Standby?On.	多軸同時駆動 Standby 状態のため実行されなかった場合
@Preset?（引数）?Er:?E?(エラーコード)	SC 本体のエラーで正常終了できなかった場合

[例]

（送信側）

sc400.theta? Preset?100	モータ theta の現在位置の値として 100 を設定します

（返信されてくる文字列）

sc400.theta>test?@Preset?100?Ok:	正常に動作を終了した場合
sc400.theta>test?@Preset?100?Er:?(エラーコード)	SC 本体のエラーで正常終了できなかった場合

SetOffset

このコマンドを送信することでモータのオフセット値を設定します。
このコマンドはメッセージ配信先として指定したモータが Busy 状態もしくは多軸同時駆動 Standby 状態のときはエラーとなり実行されません。

[引数]

モータのオフセット値

-68108813 から～68108813 までの数値文字列（＋符号の指定は不可）

[リプライ・メッセージ]

@SetOffset?（引数）?Ok:	正常に動作を終了した場合
@SetOffset?（引数）?Er:?Offset?Out?Of?Range.	オフセット値が有効範囲を超え

	ためエラーの場合
@SetOffset?（引数）?Er:?Busy.	モータが Busy 状態のため実行されなかった場合
@SetOffset?（引数）?Er:?Standby?On.	多軸同時駆動 Standby 状態のため実行されなかった場合
@SetOffset?（引数）?Er:?E?(エラーコード)	SC 本体のエラーで正常終了できなかった場合

[例]

（送信側）

sc400.theta?SetOffset?100	モータ theta にオフセット値 100 を設定します

（返信されてくる文字列）

sc400.theta>test?@SetOffset?100?Ok:	正常に動作を終了した場合
sc400.theta>test?@SetOffset?100?Er:?(エラーコード)	SC 本体のエラーで正常終了できなかった場合

GetValue

このコマンドを送信することでモータの現在位置の値を取得します。

[引数]

モータの現在位置の値の形式

0	パルス
1	パルス+オフセット
2	角度換算値
3	角度換算値＋オフセット

モータの現在位置の値の形式について詳細を知りたい場合は、SC 本体の取扱説明書にあるコマンド「RDP」をご確認ください。

[リプライ・メッセージ]

@GetValue?（引数）?（モータの現在位置の値）	データ取得が正常におこなわれた場合
@GetValue?（引数）?Er:?Bad?command?or?parameter	引数が間違っていてエラーの場合
@ GetValue?（引数）?Er:?E?(エラーコード)	SC 本体のエラーで正常終了できなかった場合

[例]

（送信側）

sc400.theta?GetValue?1	パルス+オフセットの形式でモータ theta の現在位置の値を取得します

（返信されてくる文字列）

sc400.theta>test?@GetValue? 1?10000　モータ theta の現在位置がパルス＋オフセットの形式の値 10000 として返ってきます

sc400.theta>test?@GetValue?1?Er:?E?(エラーコード)

SC 本体のエラーで正常終了できなかった場合

GetOffset

このコマンドを送信することでモータのオフセット値を取得します。

[リプライ・メッセージ]

@GetOffset?（モータのオフセット値）　データ取得が正常におこなわれた場合

@GetOffset?Er:?E?(エラーコード)　SC 本体のエラーで正常終了できなかった場合

[例]

（送信側）

sc400.theta?GetOffset　モータ theta のオフセット値を取得します

（返信されてくる文字列）

sc400.theta>test?@GetOffset?100　モータ theta のオフセット値 100 が返ってきます

sc400.theta>test?@GetOffset?Er:?E?(エラーコード)

SC 本体のエラーで正常終了できなかった場合

SetValue

このコマンドを送信することでモータの位置を絶対値で指定された値まで移動します。
このコマンドはメッセージ配信先として指定したモータが Busy 状態のときはエラーとなり実行されません。
また、多軸同時駆動 Standby 状態の場合コントローラコマンド「SyncRun」を実行するまで移動動作はおこなわれません。

[引数]

①モータの移動目標位置の絶対値モータの移動目標位置の絶対値

-68108813 から～68108813 までの数値文字列（＋符号の指定は不可）

②加減速モード

1 から 5 までの数字文字

③同期モード

0 もしくは 1 の数字文字

④速度テーブル No.

0 から 9 までの数字文字

⑤バックラッシュ補正

0 から 4 までの数字文字

⑥エンコーダ補正

0から2までの数字文字

※②～⑥の引数について詳細を知りたい場合は、SC本体の取扱説明書にあるコマンド「APS」をご確認ください。

[リプライ・メッセージ]

@SetValue?（引数）?Ok:	正常に動作を終了した場合
@SetValue?（引数）? Er:?Data?Out?Of?Range.	モータの移動目標位置（絶対値）が有効範囲を超えたためエラーの場合
@SetValue?（引数）? Er:?Bad?command?or?parameter	②～⑥の引数が有効範囲を超えたためエラーの場合
@SetValue?（引数）? Er:?SetValue?Executed?Over:?Max	多軸同時駆動 Standby 状態で SetValue コマンドを最大軸数もしくは 4 回を超えて実行したためエラーの場合
@SetValue?（引数）? Er::?Busy.	モータが Busy 状態のため実行されなかった場合
@SetValue?（引数）? Er: ? E? (エラーコード)	SC 本体のエラーで正常終了できなかった場合

[例]

（送信側）

sc400.theta?SetValue?10000?2?0?1?3?0　　モータ theta を絶対値 10000 まで移動する

（返信されてくる文字列）

sc400.theta>test?@SetValue?1?10000?2?0?1?3?0?Ok:　　正常に動作を終了した場合

sc400.theta>test?@SetValue?1?10000?2?0?1?3?0?Er:?(エラーコード)

SC本体のエラーで正常終了できなかった場合

SetValueREL

このコマンドを送信することでモータの位置を現在位置からの相対値で指定された値まで移動します。

このコマンドはメッセージ配信先として指定したモータが Busy 状態もしくは多軸同時駆動 Standby 状態のときはエラーとなり実行されません。

[引数]

①モータの移動目標位置の現在位置からの相対値

-68108813 から～68108813 までの数値文字列（＋符号の指定は不可）

②加減速モード

1から5までの数字文字

③同期モード

0もしくは1の数字文字

④速度テーブル No.

0から9までの数字文字

⑤バックラッシュ補正

0から4までの数字文字

⑥エンコーダ補正

0から2までの数字文字

※②～⑥の引数について詳細を知りたい場合は、SC本体の取扱説明書にあるコマンド「RPS」をご確認ください。

[リプライ・メッセージ]

@SetValueREL?（引数）?Ok:	正常に動作を終了した場合
@SetValueREL?（引数）? Er:?Data?Out?Of?Range.	引数のモータの移動目標位置（現在位置からの相対値）が有効範囲を超えたためエラーの場合
@SetValueREL?（引数）? Er:? Bad?command?or?parameter	②～⑥の引数が有効範囲を超えたためエラーの場合
@SetValueREL?（引数）? Er::?Busy.	モータが Busy 状態のため実行されなかった場合
@SetValueREL?（引数）?Er:?Standby?On.	多軸同時駆動 Standby 状態のため実行されなかった場合
@SetValueREL?（引数）? Er: ? E ?(エラーコード)	SC 本体のエラーで正常終了できなかった場合

[例]

（送信側）

sc400.theta?SetValueREL?100?2?0?1?3?0	モータ theta を現在位置から+の方向に 100 移動する

（返信されてくる文字列）

sc400.theta>test?@SetValueREL?1?10000?2?0?1?3?0?Ok:	正常に動作を終了した場合
sc400.theta>test?@SetValueREL?1?10000?2?0?1?3?0?Er:?(エラーコード)	SC本体のエラーで正常終了できなかった場合

Stop

このコマンドを送信するとモータが減速停止します。

[リプライ・メッセージ]

@Stop?Ok:	正常に動作を終了した場合
@Stop?Er:?E?(エラーコード)	SC 本体のエラーで正常終了できなかった場合

[例]

（送信側）

sc400.theta?Stop	モータ theta を減速停止します

（返信されてくる文字列）

sc400.theta>test?@Stop?@Ok:	正常に動作を終了した場合
sc400.theta>test?@Stop?Er:?E?(エラーコード)	SC 本体のエラーで正常終了できなかった場合

StopEmergency

このコマンドを送信するとモータが緊急停止します。

[リプライ・メッセージ]

@StopEmergency?Ok:	正常に動作を終了した場合
@StopEmergency?Er:?E?(エラーコード)	SC 本体のエラーで正常終了できなかった場合

[例]

（送信側）

sc400.theta?StopEmergency	モータ theta を緊急停止します

（返信されてくる文字列）

sc400.theta>test?@StopEmergency?@Ok:	正常に動作を終了した場合
sc400.theta>test?@StopEmergency?Er:?E?(エラーコード)	SC 本体のエラーで正常終了できなかった場合

IsBusy

このコマンドを送信することでモータが駆動しているか否かのデータを取得します。

[リプライ・メッセージ]

@Busy? 0	モータが停止状態
@Busy? 1	モータが駆動中の状態

[例]

（送信側）

sc400.theta?IsBusy	モータ theta の駆動状況を確認します

（返信されてくる文字列）

sc400.theta >test?@IsBusy?0 **モータ** theta **が停止状態の場合**

sc400.theta>test?@IsBusy?1 **モータ** theta **が駆動中の状態の場合**

GetStatus

このコマンドを送信することでモータの最新ステータスとして SC **コマンド**「STR」**の返答データをスペース区切りで返します。**

[**リプライ・メッセージ**]

@GetStatus?**動作状態（駆動動作）**?（NORG **信号**）?（ORG **信号**）?（CW **リミット信号**）?（CCW **リミット信号**）?**（振動カウント数）**?（**エラー**No. ） **正常に動作を終了した場合**

[**例**]

（送信側）

sc400.theta?GetStatus **モータ** theta **の最新ステータス値を取得します**

（返信されてくる文字列）

sc400.theta>test?@GetStatus?0?0?0?0?0?0?0 **正常に動作を終了した場合**

エンコーダコマンド

[メッセージ配信先名]

sc400.*encodername*

encodername のエンコーダに対して命令（コマンド）を送信する場合の配信先名

GetAxisNumber

このコマンドを送信することでエンコーダ名に対応する軸番号を取得します。

[例]

（送信側）

sc400.ENCtheta?GetAxisNumber	エンコーダ ENCtheta の軸番号を取得します

（返信されてくる文字列）

sc400.ENCtheta>test?@GetAxisNumb er? 1	エンコーダ ENCtheta の軸番号 1 が返ってきます

Preset

このコマンドを送信することでエンコーダの現在位置を指定値で書き換えます。
このコマンドはメッセージ配信先として指定したエンコーダと同軸のモータが Busy 状態もしくは多軸同時駆動 Standby 状態のときはエラーとなり実行されません。

[引数]

エンコーダの値

-68108813 から～68108813 までの数値文字列（＋符号の指定は不可）

[リプライ・メッセージ]

@Preset?（引数）?Ok:	正常に動作を終了した場合
@Preset?（引数）?Er:? Preset? Out?Of?Range.	引数のエンコーダの値が有効範囲を超えたためエラーの場合
@Preset?（引数）?Er:?Busy.	モータが Busy 状態のため実行されなかった場合
@Preset?（引数）?Er:?Standby?On.	多軸同時駆動 Standby 状態のため実行されなかった場合
@Preset?（引数）?Er:?E?(エラーコード)	SC 本体のエラーで正常終了できなかった場合

[例]

（送信側）

sc400.ENCtheta? Preset?100	エンコーダ ENCtheta の現在位置の値として 100 を設定します

（返信されてくる文字列）

sc400.ENCtheta>test?@Preset?100?Ok:　正常に動作を終了した場合

sc400.ENCtheta>test?@Preset?100?Er:?(エラーコード)　SC 本体のエラーで正常終了できなかった場合

GetValue

このコマンドを送信することでエンコーダの現在位置の値を取得します。

[引数]

エンコーダの現在位置の値の形式

0　パルス

1　パルス+オフセット

2　角度換算値

3　角度換算値＋オフセット

エンコーダの現在位置の値の形式について詳細を知りたい場合は、SC 本体の取扱説明書にあるコマンド「RDE」をご確認ください。

[リプライ・メッセージ]

@GetValue?（引数）?（エンコーダの現在位置の値）　データ取得が正常におこなわれた場合

@GetValue?（引数）?Er:?Bad?command?or?parameter　引数が間違っていてエラーの場合

@GetValue?（引数）?Er:?E?(エラーコード)　SC 本体のエラーで正常終了できなかった場合

[例]

（送信側）

sc400.ENCtheta?GetValue?1　パルス+オフセットの形式でエンコーダ ENCtheta の現在位置の値を取得します

（返信されてくる文字列）

sc400.ENCtheta>test?@GetValue?1?10000　エンコーダ ENCtheta の現在位置がパルス＋オフセットの形式の値10000 として返ってきます

sc400.ENCtheta>test?@GetValue?1?Er:?E?(エラーコード)　SC 本体のエラーで正常終了できなかった場合

4.4　ビームライン標準化アンケート結果

＜設問 1＞

1. 現在ご担当のビームライン・ステーションで計測および制御に使用している機器についてカテゴリーごとに列挙してください。

例： 機器名：パルスモータコントローラ　型番：PM16C-04S　メーカー：ツジ電子
　　機器名：カウンタ・タイマ　型番：974　メーカー：Ortec
　　機器名：X 線 CCD 検出器　型番：Quantum4R　メーカー：ADSC

1－1. パルスモータコントローラ・エンコーダ関連
機器名：　型番：　メーカー：

1－2. カウンタ・タイマ関連
機器名：　型番：　メーカー：

1－3. アナログ・デジタル変換関連
機器名：　型番：　メーカー：

1－4. MCA、SCA、TAC
機器名：　型番：　メーカー：

1－5. その他の計測・制御機器
機器名：　型番：　メーカー：

1-1.　パルスモーターコントローラ(PMC)、エンコーダ(EC)

機種	型番	メーカー	回答数	使用ビームライン
PMC	PM16C-02	ツジ電子	8	NE3, 構造生物, XAFS, 2A, 11A, 11B, 4A, NE5A
PMC	PM16C-04	ツジ電子	4	14A, 14B, 構造生物, NW14
PMC	PM4C-05	ツジ電子	4	14A, NW2, XAFS, 2A
PMC	SC800	神津精機	2	構造生物, NW14
PMC	SC200	神津精機	2	構造生物, NW14
PMC	SC400	神津精機	1	構造生物
PMC	MMC-2	中央精機	1	XAFS
PMC	SHOT-204	シグマ光機	1	NW2

PMC	PMCTRL	?	1	20A (PC ボードからドライバ直結)
PMC	PM2C-01	ツジ電子	1	10B
PMC	PM3C-01	ツジ電子	1	2A
PMC	PMC-3GR	神津精機	1	NE5A
EC	ND221	ハイデンハイン	2	構造生物, 2A
EC	HD261	ハイデンハイン	1	NE3
EC	ND282	ハイデンハイン	1	2A
EC	UC101	ニコン	1	20A (製造中止)
EC	ND281	ミツトヨ	1	13C
EC	LH20A	SONY	1	13C
EC	VRZ166	ハイデンハイン	1	2A

1-2. カウンタ(CO)、スケーラ(SC)、タイミングジェネレータ(TG)、DXP

機種	型番	メーカー	回答数	使用ビームライン
CO	974	ORTEC	7	XAFS, 20A, NE3, 8A, 2A, 4A
CO	994	ORTEC	3	14B, 14A, 4A
CO	995	ORTEC	1	4A
SC	4434	LeCroy	1	XAFS
SC	C257	CAEN	1	XAFS
TG	3655	Kinetics	1	XAFS
DXP	DXP-4C	XIA	1	XAFS

1-3. ADC, VFC, DMM

機種	型番	メーカー	回答数	使用ビームライン
ADC	ADA8/2	コンテック	1	XAFS
ADC	AD16-	コンテック	1	XAFS
ADC	4801A	Labo	2	構造生物, 20A
VFC	NUF-02B	ツジ電子	1	13C
DMM	2700	Keithley	1	構造生物

DMM	2701	Keithley	1	構造生物

1-4. MCA, SCA, TAC

機種	型番	メーカー	回答数	使用ビームライン
MCA	LN-6400	Labo	1	13C
MCS	LN-6500	Labo	1	13C
MCA	7700	Seiko	2	4A, NE3
MCA	MultiMax	Roentec	1	4A
MCS	MCS-PCI	Ortec	1	4A
MCS	35PLUS	Canberra	1	XAFS
MCA	XR-100CR	Amptek	1	構造生物
MCA	7800	Seiko	1	14B
MCA	MCA/9801	Labo	1	20A
MCA	5004	Aptec	1	14A

1-5. その他

機種	型番	メーカー	回答数	使用ビームライン
	6514	Keithley	1	13C
	6517A	Keithley	1	13C, 8A
	HSX-3R5	松定プレシジョン	1	8A
	TR6150	ADVANTEST	1	8A
	18011	Oriel	1	2A
	33120A	HP	1	4A
	TDS3000	Tektronix	1	4A
	C4880	浜松ホトニクス	1	4A
	6485	Keithley	1	構造生物
	Quantum##	ADSC	3	構造生物
	XFDI	PhotonicScience	1	14B

＜設問 2＞

2. 将来ご担当のビームライン・ステーションで計測および制御に使用したいと考えている機器についてカテゴリーごとに列挙してください。

例： 機器名：パルスモータコントローラ　型番：PM16C-04S　メーカー：ツジ電子
　　機器名：カウンタ・タイマ　型番：974　メーカー：Ortec
　　機器名：X 線 CCD 検出器　型番：Quantum4R　メーカー：ADSC

２－１．パルスモータコントローラ・エンコーダ関連
機器名：　型番：　メーカー：

２－２．カウンタ・タイマ関連
機器名：　型番：　メーカー：

２－３．アナログ・デジタル変換関連
機器名：　型番：　メーカー：

２－4. MCA、SCA、TAC
機器名：　型番：　メーカー：

２－５．その他の計測・制御機器
機器名：　型番：　メーカー：

2-1.　パルスモーターコントローラ(PMC)、エンコーダ(EC)		
機種	型番	メーカー
PMC	PM16C-02	ツジ電子
PMC	PM4C-05	ツジ電子
PMC	MMC-2	中央精機

2-3.　ADC, VFC, DMM		
機種	型番	メーカー
ADC		PCI バスの製品

2-4.　MCA, SCA, TAC		
機種	型番	メーカー

	WE7000 シリーズ
MCA　7600	Seiko EG&G
MCS　P7888	FAST ComTec

2-5.　その他

機種	型番	メーカー
	R6161	ADVANTEST
	9650	SI
	Marmosaic225	marUSA

＜設問 3＞

3．その他、BL 制御の標準化についてご意見があれば、記入してください。

保守、支援体制の維持が最も難しいかと思います（十分検討されているとは思いますが）。
BL 制御関係の標準化は，整備の効率や機器の保守，ユーザーへの対応を考えると，是非，必要だと思います。機器が故障したときや必要な機器が不足したときなどに相当苦労したこともありました。また，今まで PF は一般的に各ビームライン毎に整備が進められてきており，となりのステーションで当たり前のことが周りに伝わっていなかったりすることもあったと思います。ＳＰ８における光学系制御の標準化は，（課題もあるとお聞きしていますが）ユーザーとしてみたときには，実験ステーションが変わっても制御方法が同じなので楽であると感じています。
“BL 制御グループ”が、どこまで面倒を見てくれるのか、それが“ユーザー”であるステーション担当者にとっての最大の関心事である。システム構築時に仕様を相談しながらユーザーインターフェース（GUI）まで作ってもらえること、そしてアフターサービス（バグ取りは当然のこととしてステーションの状況によるソフトの修整／改造など）がしっかりしていることが、できることならば望ましい。
この際ですから、古い機器のサポートは断念して、ある程度標準化を図る必要があるでしょう。ただ、今後とも機器は新しくなるので、ライブラリ的に継続的に充実して行く必要があるでしょう。一方で、今の計算機は高性能なので、過度の標準化は避け、但し単に趣味による多様化は必要ないように、ソフト開発をお願いしたいと思います。後々の保守を考えると、出来るだけ定義ファイルをいじる程度でソフト本体はいじらないで多くのラインに適用できることが好ましいと思います。

①光学系制御および自動ステージ制御プログラムをLabVIEWで整備中。STARSとうまく統合して運用したい（ハッチ内に設置した回折計はCプログラムで制御、STARSを介して光学系と連動させて計測したいーCAMAC系で測定）。 ②居室からステーションの状況のモニターを希望（監視カメラ含む）。標準的な方法、システムを教えてほしい。
諸機器の組み合わせて，さらに実験パラメータを変えて，いろいろな測定のルーチンをくむのが難しいです。ところで，ＰＦ仕様のデータファイルの構造を決めましたか？

4．5 第１回ビームライン制御に関する打合せ資料

ビームライン制御に関する打ち合わせ
2005.4.18

ＢＬ制御システム(VSX)

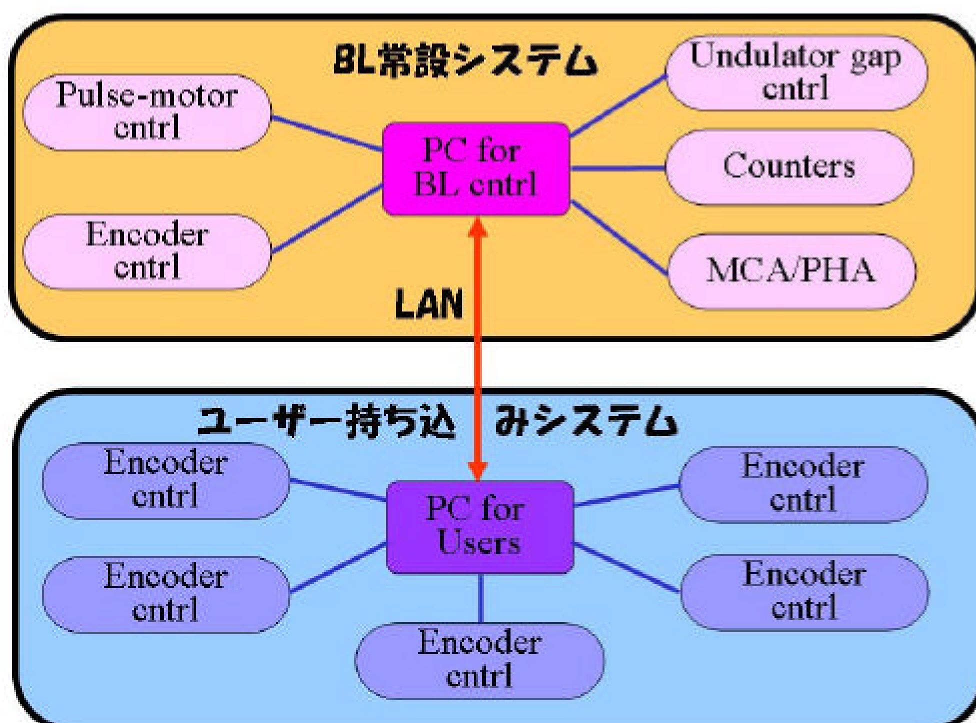
従来型１：ＢＬ担当者によるシステムの構築
従来型２：2C 3B 12A 16B 20Aにおける統一ソフト

専門家集団による系統的なシステム構築・維持管理

Up-to-dateな技術の導入
マンパワーの効率化
トラブルへに早い対応
測定システムへの助言

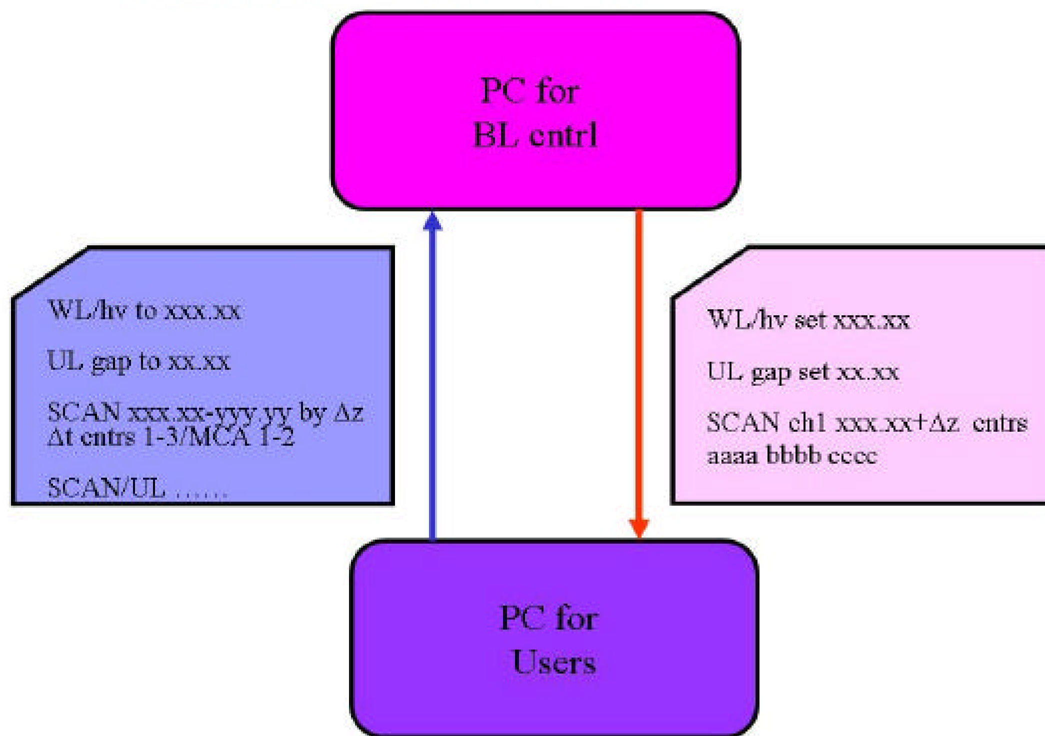
ビームライン制御に関する打ち合わせ
2005.4.18

ＢＬ制御システムの概略



ビームライン制御に関する打ち合わせ
2005.4.18

PC間のコミュニケーション



ビームライン制御に関する打ち合わせ
2005.4.18

BL制御システムの統一化

常設システムの統一化
- BL cntrl PCのソフト
- 接続機器／インターフェイス
- BL cntrl PC-User PC間のプロトコール
- User PCの標準的ソフト支援

BL cntrl PCの総括的監視システムの構築
- メンテナンス、グレードアップの効率化